Jagiellonian University

Faculty of Physics, Astronomy and Applied Computer Science

Michał Rawlik

Album number: 1051524

FID signal analysis and new DAQ system in the nEDM experiment

Master Thesis

studies programme
Experimental Physics

Supervisor:
dr hab. Jacek Zejma
Nuclear Physics Division

Cracow
July 5, 2014

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Kraków, dnia   Michał Rawlik

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Kraków, dnia   dr hab. Jacek Zejma

JAGIELLONIAN UNIVERSITY

FACULTY OF PHYSICS, ASTRONOMY AND APPLIED COMPUTER SCIENCE

Abstract

FID signal analysis and new DAQ system in the nEDM experiment

Master Thesis

by Michał Rawlik

The nEDM experiment, carried out at the Paul Scherrer Institute in Villigen, Switzerland, measures the electric dipole moment of the neutron. An important part of the set–up is the ^{199}Hg cohabiting magnetometer. A thorough research on the magnetometer FID signal analysis is presented in the first part of this thesis. A new approach to the analysis is proposed, together with routines to assess statistical and systematic uncertainties. A full, step–by–step analysis of two example data sets is demonstrated. Additionally, a software to investigate and perform the analysis is published. The second part of this thesis presents a design of a new control and data acquisition system for the nEDM experiment. Results of quantitative tests of the system's ADC, DAC and timing capabilities are presented, proving that the system is ready to be deployed in the experimental set–up.

UNIWERSYTET JAGIELLOŃSKI

WYDZIAŁ FIZYKI, ASTRONOMII I INFORMATYKI STOSOWANEJ

Streszczenie

Analiza synału FID i nowy system DAQ w eksperymencie nEDM

Praca Magisterska

Michał Rawlik

Eksperyment nEDM przeprowadzany w Instytucie Paula Scherrera, w Villigen w Szwajcarii mierzy elektryczny moment dipolowy neutronu. Ważnym elementem układu eksperymentalnego jest kohabitujący magnetometr oparty na atomach ^{199}Hg. Przedmiotem pierwszej części niniejszej pracy są rozważania na temat analizy sygnału FID pochodzącego z magnetometru. Zaproponowane jest nowe podejście do jego analizy oraz metody szacowania w niej niepewności statystycznych i systematycznych. Krok po kroku, przedstawiona jest analiza dwóch przykładowych zestawów danych. Ponadto, zaprezentowane jest oprogramowanie służące zarówno do przeprowadzania tej analizy, jak i do badania poszczególnych jej kroków. Druga część pracy prezentuje projekt nowego rozwiązania do kontroli i akwizycji danych w eksperymencie nEDM. Przedstawione są wyniki ilościowych testów dotyczących możliwości czasowych, ADC i DAC, które dowodzą gotowości systemu do wdrożenia w układ eksperymentalny.

Acknowledgements

Most of all, I would like to thank Dr. Jacek Zejma. Not only for being an invaluable advisor, but also for an unprecedented friendliness during the two years, when we shared an office. I am also very grateful to all members of Prof. Kazimierz Bodek's research group, including himself, for numerous valuable remarks on my work. Besides, I would like to thank the researchers working by the nEDM experiment in the Paul Scherrer Institute — I have learned a lot from you during my stays there. In particular, I would like to mention Dr. Georg Bison for his explanations and apt remarks that put me on the right course with the work.

Contents

List of Figures **ix**

1 Introduction **1**

2 The nEDM experiment in PSI **3**
- 2.1 Ramsey's method of oscillatory fields . . . 3
- 2.2 Principle of measurement . . . 4
- 2.3 Overview of the experimental apparatus . . . 5
  - 2.3.1 Delivery, storage and detection of UCNs . . . 6
  - 2.3.2 Generation of magnetic and electric fields . . . 6
  - 2.3.3 Stability and measurement of B_0 field . . . 7
- 2.4 Schedule of a measurement cycle . . . 7

3 Analysis of the ^{199}Hg comagnetometer FID signal **9**
- 3.1 Principle of operation . . . 9
- 3.2 Hardware structure of the comagnetometer . . . 10
- 3.3 Properties of the signal . . . 10
- 3.4 The problem of frequency estimation . . . 12
- 3.5 The programming toolkit . . . 13
- 3.6 Characterisation of a basic method . . . 14
  - 3.6.1 Simulation of a signal . . . 14
  - 3.6.2 Basic statistical correctness test . . . 14
- 3.7 Accuracy and precision estimation in presence of frequency drifts . . . 15
- 3.8 The filter . . . 19
- 3.9 Two windows method . . . 21
  - 3.9.1 The original method . . . 21
  - 3.9.2 Implementation . . . 22
  - 3.9.3 Location of ϕ_2 . . . 24
  - 3.9.4 Size of windows . . . 25
  - 3.9.5 Influence of the filter . . . 27
  - 3.9.6 Conclusion . . . 29
- 3.10 *Point-by-point* phase analysis method . . . 29
  - 3.10.1 Description . . . 29
  - 3.10.2 Case of variable frequency . . . 32
  - 3.10.3 Summary . . . 34
- 3.11 Windowed phase analysis method . . . 35
  - 3.11.1 Motivation . . . 35

3.11.2 Description . . . 35
3.11.3 Adaptive linear extrapolation . . . 36
3.11.4 Performance . . . 38
3.11.5 Summary . . . 39
3.12 Oscillations of the signal envelope . . . 40
3.13 Data analysis . . . 41
3.13.1 Proposed scheme . . . 41
3.13.2 Analysis of run with short τ – run 7650 . . . 43
3.13.3 Analysis of run with long τ – run 4658 . . . 47

4 Design of a PXI–based DAQ and control system 51
4.1 Introduction . . . 51
4.2 The daqBoard . . . 52
4.3 PXI system . . . 53
4.4 Software design . . . 56
4.4.1 Technologies used . . . 56
4.4.2 Structure of the software . . . 57
4.4.3 Assurance of phase coherence of ELF pulses . . . 58
4.4.4 ADC readout handling . . . 58
4.4.5 Triggering . . . 59
4.5 Interface of the system . . . 59
4.5.1 Goals . . . 59
4.5.2 Commands . . . 60
4.5.3 Examples . . . 61
4.6 Tests . . . 64
4.6.1 Introduction . . . 64
4.6.2 Methodology . . . 64
4.6.3 Synchronisation test . . . 65
4.6.4 Comparison of ADC modules . . . 67
4.6.5 Timing accuracy of TTL pulses . . . 69
4.6.6 On–site test . . . 70
4.6.7 Summary . . . 71

5 Summary 73

A The `hg_toolkit` 75
A.1 The choice of the programming language . . . 75
A.2 Structure of the toolkit . . . 76

B Full list of implemented PXI commands 79

Bibliography 81

List of Figures

1.1 Effects of action of parity–reversal operator $\mathcal{P}$ and time–reversal operator $\mathcal{T}$ on a system. 1
1.2 Experimental upper limits on neutron electric dipole moment. 2

2.1 Ramsey's method of separated oscillatory fields. 4
2.2 Scheme of the nEDM experimental set–up. 5

3.1 Energy levels of an ^{199}Hg atom in magnetic field. 10
3.2 An example signal from the ^{199}Hg comagnetometer. 11
3.3 Explanation of terms accuracy and precision. 13
3.4 Distribution of Δ over 5000 signals, as given by the basic method. 15
3.5 Several example frequency drifts. 16
3.6 Diagram showing methodology of assessing precision and accuracy of average frequency estimation. 17
3.7 Distributions of accuracy and precision over random drifts for *the basic method*. 18
3.8 Estimated precision and accuracy of the basic method of average frequency estimation. 18
3.9 The frequency response of the implemented Butterworth filter and the hardware analogue filter. 19
3.10 Three power spectra: measured signal, transmission of band–pass filter used in set-up and filtered signal. 20
3.11 Effect of applying band–pass filter to white noise in time and frequency domain. 20
3.12 Signal with signal-to-noise ratio ≈ 5 with zero-crossings marked. 24
3.13 Explanation of possible mistake at the beginning of signal when counting periods. 24
3.14 Accuracy and precision of two variants of *two windows method*: with phase fitted at the beginning and at the end of the second window. . . . 25
3.15 ccuracy and precision of the two windows method versus size of the second window, $\tau \approx 120\,\mathrm{s}$. 26
3.16 Accuracy and precision of the two windows method versus size of the second window, $\tau \approx 40\,\mathrm{s}$. 27
3.17 Simulation results concerning influence of the filter on the *two windows* method. 28
3.18 Accuracy and precision of the *two windows* method used to analyse a filtered signal, in function of a mismatch between frequency of the signal and central frequency of the filter. 28
3.19 Close–up of point-by-point phase analysis method. 31

3.20 Basic statistical correctness test of the *point-by-point* method with straight line being fitted to whole *cumulative phase* array. 31
3.21 Basic statistical correctness test of the *point-by-point* method with straight lines fitted in two 15 s windows at ends of the signal. 31
3.22 Comparison of the two windows method and analogous one based on analysis of the *cumulative phase* array. 32
3.23 Precision and accuracy of *point-by-point phase analysis* method in idealised conditions. 33
3.24 Precision and accuracy of *point-by-point phase analysis* method in more realistic conditions. 34
3.25 Adaptive linear extrapolation of phase method. 37
3.26 Adaptive linear extrapolation of phase method — zoom. 37
3.27 Reduced χ^2 of fit in the second window in function of size of the window. 38
3.28 Reduced χ^2 of fit in the first window in function of size of the window. . 39
3.29 Oscillations of the envelope of the comagnetometer signal, run 7650. . . . 40
3.30 Deviation of amplitude of the signal from sum–of–two–exponents decay throughout a cycle, for 100 cycles. 41
3.31 Power spectrum of data from Fig.3.30. 41
3.32 Histogram of the comagnetometer signal–to–noise ratio for all cycles of run 7650. 43
3.33 Histogram of relaxation times of the comagnetometer signal for all cycles of run 7650. 43
3.34 Cycle-to-cycle frequency change in run 7650. 44
3.35 Accuracy and precision of average precession frequency estimation, as a function of size of the second window. 45
3.36 Accuracy and precision of the average frequency estimation, as a function of a mismatch between the precession frequency and the central frequency of the filter. 45
3.37 Average precession frequency in each cycle of run 7650, analysed with optimised method. 46
3.38 Average precession frequency in first 50 cycles of the run 7650, as estimated with four methods. 47
3.39 Histogram of the comagnetometer signal–to–noise ratio for all cycles of run 4658. 47
3.40 Histogram of relaxation times of the comagnetometer signal for all cycles of run 4658. 47
3.41 Cycle-to-cycle frequency change in run 4658. 48
3.42 Accuracy and precision of average precession frequency estimation, as a function of size of the second window. 49
3.43 Accuracy and precision of average precession frequency estimation, as a function of mismatch between precession frequency and central one of the filter. 49
3.44 Average precession frequency in each cycle of run 4658, analysed with optimised method. 50
3.45 Average precession frequency in first 50 cycles of the run 4658, as estimated with four methods. 50

4.1 An example *cycle* as defined in the *AcqEDM* program. 52

4.2 A photograph of the third revision of the daqBoard. 53
4.3 Front of 8-Slot PXI Chassis. 54
4.4 PXI Express timing and triggering buses. 55
4.5 A fragment of LabVIEW G language source–code. 56
4.6 Method used to test frequency stability. 65
4.7 Results of the frequency stability measurement of the PXI system — big drift 66
4.8 Results of frequency stability measurement of the PXI system — small drift 66
4.9 National Instruments PXI-4461 module 67
4.10 National Instruments PXI-5922 module 67
4.11 Results of frequency stability measurement of NI PXI-4461 module . . . 68
4.12 Results of frequency stability measurement of NI PXI-5922 module . . . 69
4.13 Result of TTL timing accuracy measurement. 70
4.14 Recorded ^{199}Hg comagnetometer signal in a cycle performed with the PXI system. 71

A.1 A list of files that are bundled with the `hgtoolkit`. 77

Chapter 1

Introduction

Measurements of electric dipole moments of particles give insight into the most fundamental symmetries of nature. The reason is that any non-zero electric dipole moment (EDM) of a system with a non-degenerate ground state violates both parity $\mathcal{P}$ and time–reversal $\mathcal{T}$ symmetries. As seen in Fig. 1.1, an action of either $\mathcal{P}$ or $\mathcal{T}$ operator results in a different system, if electric dipole moment $\vec{d} \neq 0$. It means, that either $\vec{d} = 0$ and both symmetries are conserved, or $\vec{d} \neq 0$ and both symmetries are violated by interactions constituting the system. Additionally, if conservation of a combined $\mathcal{CPT}$ symmetry ($\mathcal{C}$ being particle–antiparticle exchange) is assumed, a non-zero EDM implies $\mathcal{CP}$ violation. Levels at which those symmetries are broken are crucial parameters in many physical theories, in particular those attempting to explain some famous, yet unexplained phenomena, such as baryon–antibaryon asymmetry in the Universe.

As measurement of an EDM of a charged system presents a huge experimental challange, neutral ones are given more attention. Most of all atoms and neutrons. Because atoms are compound systems, it is the neutron electron dipole moment what gives the clearest insight into the elementary symmetry violation processes. It is thus of no surprise, that experimentalists have been performing measurements of the neuron EDM with ever increasing precision for over 50 years, which is summed up in Fig. 1.2.

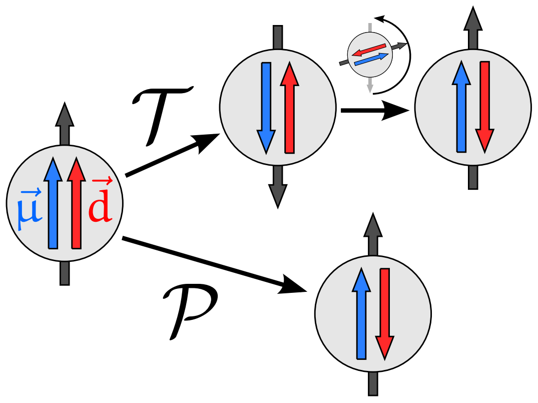


FIGURE 1.1: *Effects of action of parity operator $\mathcal{P}$ and time–reversal operator $\mathcal{T}$ on a system. Electric $\vec{d}$ and magnetic $\vec{\mu}$ dipole moments are shown.*

Recently, all neutron EDM measurements are based on a magnetic resonance of stored neutrons. They are possible thanks to *ultracold neutrons* (UCNs) — neutrons with energies below 300 neV. With such low energy, UCNs have de Brogile wave length of 50 nm, which far exceeds distance between atoms in condensed matter. As a result, rather than interact with a single nucleus, they feel an effective potential formed by many. It turns out, that for some materials this potential is highly repulsive, therefore allowing to store UCNs in containers made out of them.

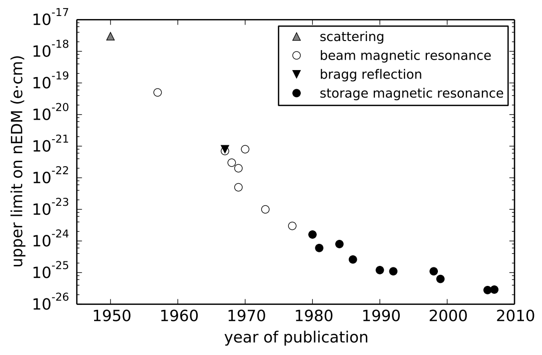


FIGURE 1.2: *Experimental upper limits on neutron electric dipole moment, obtained with four different methods of measurement.*

The currently most advanced set-up dedicated to the neutron EDM measurement is located at the Paul Scherrer Institute in Switzerland, where the most powerful in the world ultracold neutron source is being developed. The principle of this measurement, as well as the experimental set-up, are briefly described in Chapter 2.

In Chapter 3 an analysis of the signal generated by the ^{199}*Hg comagnetometer* is presented. This system is a distinguish feature of the state–of–the–art neutron EDM experiments. It measures magnetic field in the volume where UCNs are stored with a precision of the order of 10^{-6}. A thorough research was conducted concerning analysis of the comagnetometer signal. Various methods of the analysis are tested, and ways to estimate both statistical and systematic uncertainties are presented. The study is concluded by a detailed algorithm of the analysis, together with study of two sets of data.

Chapter 4 is dedicated to the central part of the data acquisition system – *the micro–timer*. It is a device that is mainly responsible for the control of the set-up with precise timing. It also collects and makes first analysis of the comagnetometer signal. The current solution is presented and a new, completely different one, proposed. Design of the new solution is described, from both hardware and software side. Finally, results of tests of the new solution are presented.

Additionally, two technical appendices are included at the end of this work. First one, appx. A, is a description of a software developed for analysis of the *mercury comagnetometer* signal. The second, appx. B, is a documentation listing all commands of interface implemented in the new *micro-timer* solution.

Chapter 2

The nEDM experiment in PSI

2.1 Ramsey's method of oscillatory fields

When a system possessing a magnetic moment $\vec{\mu}$ is placed in a magnetic field $\vec{B}$, a torque is exerted upon it:

$$\vec{\Gamma} = \vec{\mu} \times \vec{B} \,. \tag{2.1}$$

This causes the magnetic moment to precess around the magnetic field vector with a frequency called the *Larmor frequency*:

$$f_L = \frac{\mu}{Jh} B = \frac{1}{h} \gamma B \,, \tag{2.2}$$

where J is a magnitude of angular momentum of the system, and γ is called the *gyromagnetic ratio*; h is the Planck constant.

An interesting behaviour arises, when an additional small component is added to the magnetic field, one oscillating with system's Larmor frequency in a plane perpendicular to the static component $\vec{B}_0$. The oscillating component I shall call B_{ELF}, for Extreme Low Frequency, as it is only 30 Hz for neutrons in a 1 µT field in the nEDM experiment. What then happens, is that the spin of the system slowly oscillates between being parallel and antiparallel to $\vec{B}_0$, while still precessing around it with its Larmor frequency. It can be visualised, as end of the spin vector tracing a spiral on a sphere (which is called the *Bloch sphere* in a quantum case). Speed of the parallel–antiparallel oscillation depends on the magnitude of the $\vec{B}_{\mathrm{ELF}}$ field.

This phenomenon is used by the magnetic resonance techniques. Imagine a polarised sample in a magnetic field $\vec{B}_0$, with spins parallel to it. A pulse of a B_{ELF} field is applied with duration tuned such, that it stops exactly halfway in the parallel–antiparallel transition. The spins are left perpendicular to $\vec{B}_0$, continuously precessing around it. Such an ELF pulse is called *a $\frac{\pi}{2}$ pulse.*

At the foundation of *Ramsey's method of separated oscillatory fields* lie two $\frac{\pi}{2}$ pulses. The method is illustrated in Fig. 2.1. (1.) A polarised sample is placed in a magnetic field $\vec{B}_0$. (2.) Then a $\frac{\pi}{2}$ pulse, gated from a constant frequency generator, rotates spins of the sample perpendicularly to $\vec{B}_0$. (3.) Now the sample is allowed to *freely precess* in the $\vec{B}_0$ field for a substantial amount of time. (4.) Afterwards another ELF pulse is applied, which has same duration as the first one and is in–phase with it. (4.a) If frequency of the generator exactly matches the Larmor frequency of the sample, polarisation of the sample is rotated by another $\frac{\pi}{2}$, leaving it polarised antiparallel to $\vec{B}_0$. However, if the generator is even slightly detuned, then during the free precession a phase difference will build up between the generator and rotation of the spins in the sample. This will cause the polarisation not to rotate fully by $\frac{\pi}{2}$. (4.b) In particular, if the difference is π when the second pulse is gated on, the polarisation of the sample will be rotated by $\frac{\pi}{2}$ in the other direction, leaving the sample polarised parallel to $\vec{B}_0$. Sufficiently long period of the free precession causes final polarisation of the sample to be extremely sensitive to changes in frequency of the generator. This is the most precise method of Larmor frequency measurement.

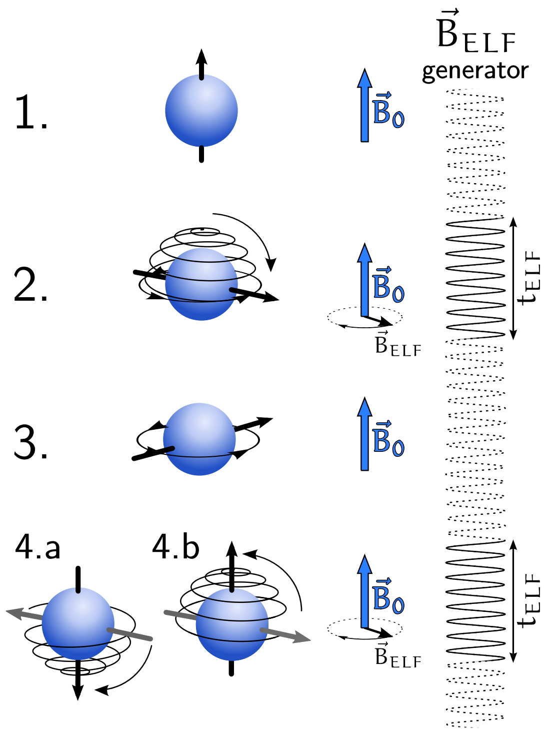


FIGURE 2.1: *Ramsey's method of separated oscillatory fields, explained in text.*

2.2 Principle of measurement

The Larmor frequency of a neutron placed in a combination of electric and magnetic fields is given by:

$$hf_L = 2\left|\vec{\mu}_n \cdot \vec{B}_0 + \vec{d}_n \cdot \vec{E}\right| , \tag{2.3}$$

where $\vec{\mu}_n$ and $\vec{d}_n$ are magnetic and electric dipole moments of neutron, respectively. If two measurements are done: with $\vec{B}_0$ and $\vec{E}$ fields parallel ($\uparrow\uparrow$) and antiparallel ($\uparrow\downarrow$), there will be a difference in Larmor frequencies in the two cases:

$$h\left(f_L^{\uparrow\uparrow} - f_L^{\uparrow\downarrow}\right) = 2\mu_n\left(B_0^{\uparrow\uparrow} - B_0^{\uparrow\downarrow}\right) + 2d_n\left(E^{\uparrow\uparrow} + E^{\uparrow\downarrow}\right) \approx 2\mu_n\left(B_0^{\uparrow\uparrow} - B_0^{\uparrow\downarrow}\right) + 4d_nE \tag{2.4}$$

Then neutron electric dipole moment can be expressed as:

$$d_n = \frac{h}{4E}\,\Delta f_L - \frac{\mu_n}{2E}\,\Delta B_0\,. \tag{2.5}$$

The principle of the measurement is to determine the Larmor frequencies, $f_L^{\uparrow\uparrow}$ and $f_L^{\uparrow\downarrow}$, in the two arrangements of magnetic and electric fields. Then Eq. 2.5 may be used to derive the neutron electric dipole moment.

Two important experimental facts follow immediately from Eq. 2.5. First is that a difference in the magnitude of the magnetic field between the two measurements directly translates onto a systematic error. The other fact is that sensitivity is proportional to square of electric field magnitude:

$$\sigma\left(d_n\right) \propto \frac{1}{E^2}\;. \tag{2.6}$$

Therefore, the electric field is desired to be as strong as possible, whereas the magnetic field is desired to be very stable, to make ΔB_0 small, as well as precisely measured, to know the value of ΔB_0.

2.3 Overview of the experimental apparatus

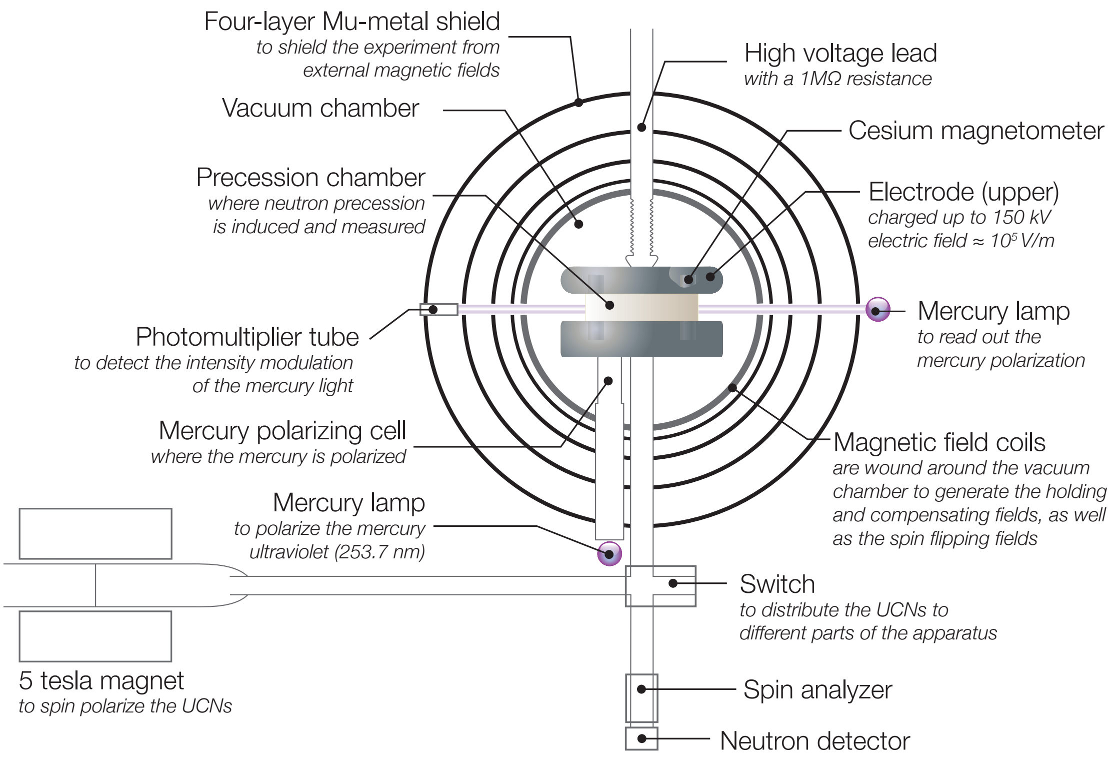

Four-layer Mu-metal shield
to shield the experiment from external magnetic fields
Vacuum chamber
Precession chamber
where neutron precession is induced and measured
Photomultiplier tube
to detect the intensity modulation of the mercury light
Mercury polarizing cell
where the mercury is polarized
Mercury lamp
to polarize the mercury ultraviolet (253.7 nm)
5 tesla magnet
to spin polarize the UCNs
High voltage lead
with a 1MΩ resistance
Cesium magnetometer
Electrode (upper)
charged up to 150 kV electric field ≈ 10⁵ V/m
Mercury lamp
to read out the mercury polarization
Magnetic field coils
are wound around the vacuum chamber to generate the holding and compensating fields, as well as the spin flipping fields
Switch
to distribute the UCNs to different parts of the apparatus
Spin analyzer
Neutron detector

FIGURE 2.2: *Scheme of the nEDM experimental set–up, described in text. Source: [1].*

A detailed scheme of the experimental set–up is presented in Fig. 2.2. Its components may be divided into three groups: the first one is responsible for delivery, storage and detection of ultracold neutrons, the second one creates the magnetic and the electric fields, and the third one ensures stability and measurement of the B_0 field. They will be briefly described below.

2.3.1 Delivery, storage and detection of UCNs

Network of UCN guides allows transport of the UCNs between various areas of set–up. In particular, it connects the UCN source to the apparatus. The guides are glass tubes covered from inside with nickel (85%) and molybdenum (15%) — a material highly repulsive for UCNs.

5 tesla superconducting magnet polarises UCNs incoming from the source.

Switch is a complex, mechanically actuated device that can route the flow of the UCNs between the source, the precession chamber and the detector.

Precession chamber, also called **the storage chamber**, is a volume where the UCNs are stored during the period of *free precession*. It is enclosed with diamond–like carbon coated electrodes from top and bottom. Sides are made up by a ring coated from the inside with *deuterated polystyrene* (another material with a very good UCNs repulsion properties).

UCN shutter is a gate valve that closes the passage between the precession chamber and the UCN guide below it.

Vacuum system provides UCN–friendly low–pressure environment.

Neutron detection system is composed of the **spin analyser**, allowing only one spin–state of UCNs through, and the **neutron detector** itself. The neutrons are detected with a ^{6}Li enriched glass scintillators with use of the $n + {}^6\mathrm{Li} \to {}^3\mathrm{H} + {}^4\mathrm{He}$ reaction.

2.3.2 Generation of magnetic and electric fields

Magnetic field coils are in a large number wound around the apparatus. They create the B_0 field and ensure its uniformity. They also generate oscillating $\vec{B}_{\mathrm{ELF}}$ pulses.

Two electrodes form a big capacitor creating electric field of 10 000 V/m. At the same time they enclose the precession chamber.

2.3.3 Stability and measurement of B_0 field

Four layer μ–metal shield is made of an alloy with a very high magnetic permeability. It shields the inside of the apparatus from external magnetic field and its fluctuations.

Caesium magnetometers provide constant measurement of magnetic field directly above and below the precession chamber.

Mercury comagnetometer, is mainly composed of: **two mercury lamps**, **the polarising cell** and a **photomultiplier**. It measures magnetic field during the free precession period directly inside the storage chamber. The Mercury comagnetometer is described in detail in the next chapter, in Sec. 3.2.

2.4 Schedule of a measurement cycle

In the nEDM experiment standard, data taking, measurements are performed by repeating a set of defined actions. One such iteration is called *a cycle*, and a group of consecutive cycles forms *a run*. Below a schedule of a typical cycle is presented.

1. The system is prepared: the switch is set to guide UCNs from the source into the storage chamber, the UCN shutter is opened. ^{199}Hg gas is polarised in the polarising cell.
2. A pulse from the UCNs source, marking its readiness, is awaited.
3. The UCN source delivers UCNs gas, which fills the apparatus up to the precession chamber.
4. The UCN shutter is closed, trapping UCNs inside the precession chamber.
5. The precession chamber is filled with a polarised ^{199}Hg gas.
6. An ELF pulse, tuned to rotate spins of the ^{199}Hg atoms by $\frac{\pi}{2}$, is applied.
7. The first ELF pulse, tuned to rotate spins of the UCNs by $\frac{\pi}{2}$, is applied.
8. System waits $120-180$ s. During this time UCNs and mercury atoms freely precess in the chamber. Precession of ^{199}Hg atoms is read out live.
9. The UCN switch is set to route UCNs from the precession chamber into the detector.
10. The second ELF pulse for UCNs is applied.
11. UCN shutter is opened, allowing UCNs to fall down to the detector.
12. Vacuum system is arranged to pump out remaining ^{199}Hg gas.

Chapter 3

Analysis of the ^{199}Hg comagnetometer FID signal

3.1 Principle of operation

The purpose of the ^{199}Hg comagnetometer is to measure **the average magnetic field in the same volume the UCNs occupy during the free precession**. Hence the name, *comagnetometer*, which is short for *cohabiting magnetometer*. The measurement relies on the phenomenon of Larmor precession of mercury atoms, which is probed live by beam of polarised light.

In the first step the atoms are polarised by an optical pumping. The $^{199}_{80}$Hg isotope in its ground state has no electronic angular momentum. Its total angular momentum, $F = 1/2$, comes from a single unpaired neutron in the nucleus. In presence of a magnetic field, the ground state $6^1\mathrm{S}_0$ splits into two stable Zeeman levels ($m_F = \pm 1/2$), occupied equally in an unpolarised group of atoms. In order to achieve polarisation, an excess in a population of one of the states has to be produced. The procedure (illustrated in Fig. 3.1) starts by optically pumping the atoms from the lower Zeeman state into the state with a higher electronic angular momentum $6^3\mathrm{P}_1$. The pumping light is generated by a ^{204}Hg lamp and is then circularly polarised (σ^+). The spectrum and the polarisation of the light make only the one chosen transition possible (see Fig. 3.1). Once excited to the $6^3\mathrm{P}_1$ state, the atoms decay back to $6^1\mathrm{S}_0$ state, but with a probability 1/3 they go into the higher Zeeman state $m_F = +1/2$. Those which decayed into the $m_F = -1/2$ state are pumped again. Eventually, most atoms are in the higher Zeeman state ($6^1\mathrm{S}_0, m_F = +1/2$).

The spins of such produced sample of mercury atoms are then rotated with a $\frac{\pi}{2}$ pulse and a coherent Larmor precession is observed. Dehmelt [2] showed, that for precessing atoms the probability of an excitation through a photon absorption is proportional to $\sin(2\pi f_L t)$, where f_L is the Larmor frequency. Therefore, when after the pulse a

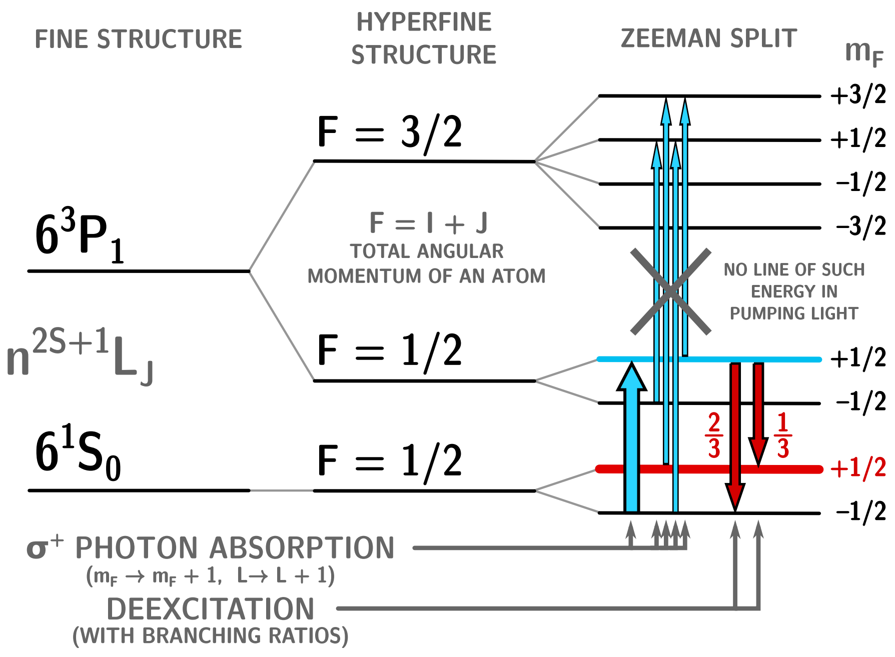


FIGURE 3.1: *Energy levels of an ^{199}Hg atom in magnetic field and optical pumping process. A polarised (σ^+) light may only cause a transition from* $(6^1\mathrm{S}_0, m_F = -1/2)$ *into* $(6^3\mathrm{P}_1, m_F = +1/2)$ *state. From there the atom deexcites back to the* $6^1\mathrm{S}_0$ *state with,* m_F *either* $-1/2$ *or* $+1/2$. *From the former it is pumped again, in the latter it stays. Effectively all atoms are pumped to* $(6^1\mathrm{S}_0, m_F = +1/2)$ *state.*

circularly polarised light shines through the gas, its transmission oscillates with the Larmor frequency. The frequency is proportional to the value of the magnetic field the gas is in, effectively making such system a magnetometer.

3.2 Hardware structure of the comagnetometer

Subsequent parts of the ^{199}Hg comagnetometer are shown in apparatus schematics in Fig. 2.2. *The source*, usually a heated mercury oxide powder, fills the *polarising cell* with ^{199}Hg atoms. There, the *mercury lamp* pumps the gas, up to 70% polarisation. In a appropriate time, a valve releases the polarised gas into the precession chamber. The second mercury lamp shines through the chamber onto the photomultiplier. When a $\frac{\pi}{2}$ pulse induces the precession, the light absorption starts to oscillate. Voltage on the photomultiplier is the directly measured observable.

3.3 Properties of the signal

In general, a signal created by a precessing system is called a *Free Induction Decay* (FID) signal. An example FID signal, from the ^{199}Hg comagnetometer, is shown in Fig. 3.2.

In an ideal case FID signals are an exponentially damped sine waves:

$$y(t) = Ae^{-t/\tau} \sin(2\pi f t + \phi_0)\,. \tag{3.1}$$

τ, called a *relaxation time*, is an important parameter in the comagnetometer. If it is much shorter than the duration of the UCNs free precession, then the amplitude of the FID signal by the end of the measurement is very small. This effectively worsens the precision of the comagnetometer.

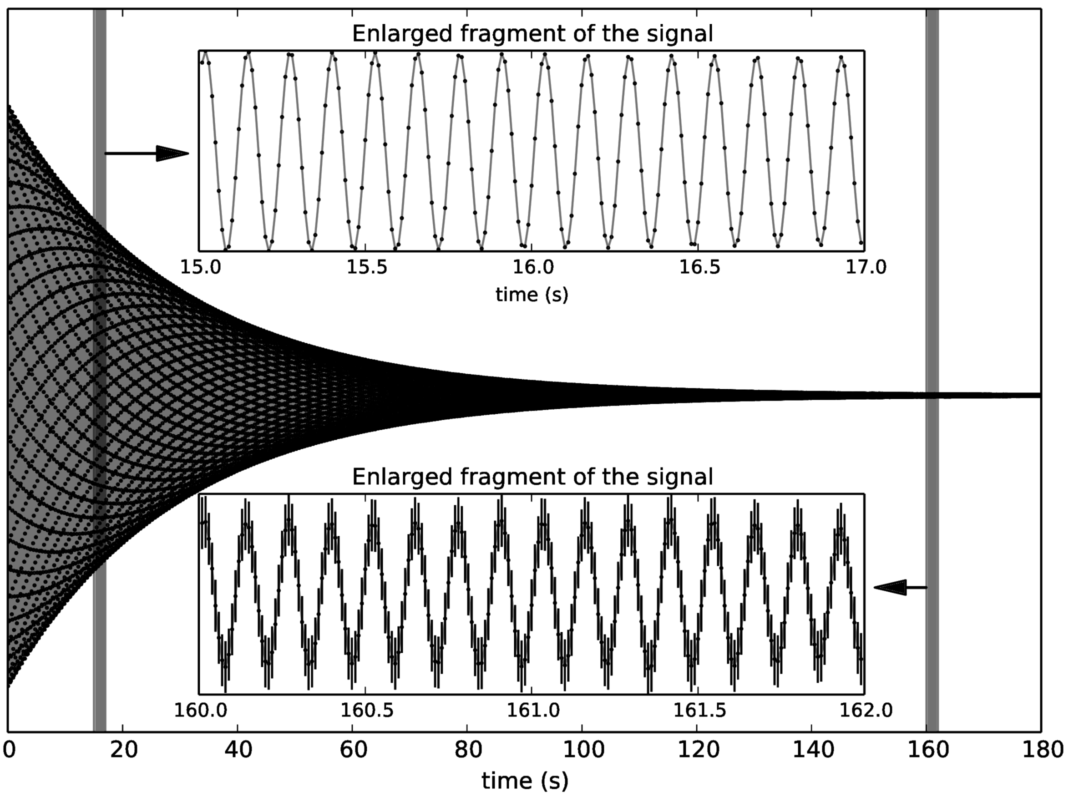


FIGURE 3.2: *An example signal from the* ^{199}*Hg comagnetometer — voltage on a photomultiplier versus time. Insets show enlarged, depicted by vertical grey strips, fragments of the signal. Actually the raw voltage does not look so smooth, but acquisition hardware filters it with a narrow band–pass filter. Signal comes from the cycle 23 of the run 7650 taken on 2013-08-16.*

It was shown by Fertl [3] that actually, due to numerous physical effects happening in the storage chamber (such as an absorption of the mercury atoms by the walls), the signal manifests two different relaxation times:

$$y(t) = \left(A_1 e^{-t/\tau_1} + A_2 e^{-t/\tau_2}\right) \sin(2\pi f t + \phi_0)\,. \tag{3.2}$$

One is usually very short (ca. 10 s), while the other ranges from 30 s, which is considered very poor, to 200 s, considered excellent. Only the value of the larger of the two relaxation times determines the amplitude of the signal by the end of the free precession. Therefore, in cases when the relaxation time is given for only general informative reasons, only the larger τ is given.

Another important property of the signal is the *signal–to–noise* ratio (SNR). The noise of the system is determined at the time just before the polarised mercury gas is filled into the precession chamber. Then there is no precession signal, so recorded voltage on the photomultiplier is a pure noise. SNR of the comagnetometer is usually in the range of hundreds.

3.4 The problem of frequency estimation

The main problem in an analysis of the comagnetometer data can be stated as finding an average frequency of the signal measured during the free precession of the UCNs. Normally, to estimate the frequency of the signal, a model described with Eq. 3.2 should be fitted to the experimental data. Unfortunately, this generic approach does not work, because the magnetic field, and hence frequency of the signal, usually changes during a cycle. The magnetic field fluctuations are of the order of a few pT per cycle, which corresponds to only a 10^{-4} relative stability. At the same time, the comagnetometer can easily reach a 10^{-5} relative precision. Therefore, the assumption about constant frequency of the signal is not fulfilled and possible fluctuations have to be incorporated in the assumed model. Unfortunately, the fluctuations may, in general, have form of any function of time $f(t)$, having thus infinite number of degrees of freedom. This number has to be reduced to just a few, which may be done in various manners.

Magnetic field fluctuations have usually a form of a slow drift. One might thus argue, that a good approach would be to expand the frequency dependence in a Taylor series and fit such constructed model. This attempt fails, though, because value of the signal at a given time is dominated by its phase and phase is **an integral of frequency**. Therefore, even an infinitesimal disagreement between the actual change and one modelled, given enough time, will result in an arbitrarily big change in the signal. Hence, other possibilities have to be explored. The main purpose of this chapter is to investigate these other ways, often called *methods*, and decide which are superior to others.

Two quantities will be used to quantitatively compare *methods* of average frequency estimation: *accuracy* and *precision*, both explained graphically in Fig. 3.3. *Precision* corresponds to statistical uncertainty, whereas *accuracy* is connected with systematic effects.

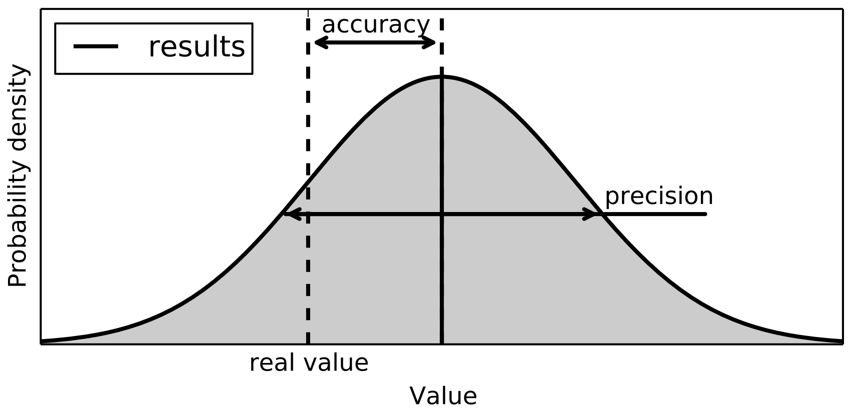


FIGURE 3.3: *Explanation of terms* ***accuracy*** *and* ***precision****. Precision is how repeatable an action is and is thus connected to statistical uncertainty. Accuracy, on the other hand, is how well action's results are centred around real value and therefore is a measure of systematic bias.*

3.5 The programming toolkit

A thorough analysis of the comagnetometer data is not straightforward and requires a fairly large amount of computer code. Furthermore, many questions arise around the analysis, including fundamental one about the method of handling the frequency drifts. These questions cannot be answered without a tool to perform simulations, that would seamlessly combine with the analysis software.

In answer to those needs a toolkit, called simply `hg_toolkit`, was created for both simulation and analysis of the comagnetometer data. It is, among other things, capable of:

- Simulating the comagnetometer signal with various parameters, including arbitrary frequency drift function $f(t)$.
- Passing simulated signal through a digital filter and testing its influence.
- Performing thorough tests of different *methods* of the signal analysis.
- Optimising parameters of different *methods* of analysis.
- Determining cases where *methods* fail to converge.
- Estimating possible systematic effects due to magnetic field drifts.
- Testing correctness of statistical uncertainty estimation.
- Testing influence of experimental conditions on precision of the comagnetometer.

In the following sections the `hg_toolkit` will be presented by fruits of its work. The technical aspects of the toolkit are discussed in Appendix A.

3.6 Characterisation of a basic method

The process of characterisation of a given *method* of comagnetometer signal analysis is explained by an example. The subject *method* will be the already mentioned most basic one, which attempts to fit to whole signal a following model:

$$y(t) = \left(A_1 e^{t/\tau_1} + A_2 e^{t/\tau_2}\right) sin\left(2\pi f t + \phi_0\right) \tag{3.3}$$

and considers optimal f yielded by the fit to be the average frequency estimate.

3.6.1 Simulation of a signal

The first step is simulation of a signal with *a priori* known values of parameters:

frequency $\mathbf{f(t)}$. In the basic model $f(t) = \text{const.}$

amplitude $\mathbf{A(t)}$. In the basic model $A(t) = A_1 e^{t/\tau_1} + A_2 e^{t/\tau_2}$

initial phase ϕ_0, which is random.

noise type and its standard deviation $\sigma(\mathbf{t})$. In the basic model white noise with a constant amplitude is assumed.

Given those parameters, the phase $\phi(t)$ is found analytically as an integral of $f(t)$ with boundary condition $\phi(0)$:

$$\phi(t) = \phi_0 + \int_0^t f(t')\mathrm{d}t' \, . \tag{3.4}$$

The signal is than calculated in the following way:

$$y(t) = A(t) \sin\phi(t) + \mathrm{randn}(0, \sigma(t)) \, , \tag{3.5}$$

where $\mathrm{randn}(0, \sigma(t))$ is a random number from normal distribution centred at 0 and with width σ.

The average frequency of the generated signal is found **analytically**:

$$\overline{f} = \int_0^T f(t)\mathrm{d}t \tag{3.6}$$

3.6.2 Basic statistical correctness test

Every method estimating the average frequency of the signal is expected to yield two values:

$E\left(\overline{f}\right)$ — an estimator of average frequency

$s\left(E\left(\overline{f}\right)\right)$ — an estimator of standard deviation of the above estimate, i.e. precision estimate

Let's define Δ as an error of estimated average frequency normalised to the precision estimator:

$$\Delta := \frac{E\left(\overline{f}\right) - \overline{f}}{s\left(E\left(\overline{f}\right)\right)} \tag{3.7}$$

If the *method* is statistically correct, then its use in a number of randomly generated signals will give Δ according to a normal distribution with the width equal to 1. Abnormalities in distributions of Δ have a clear interpretation: centre not in zero indicates bias; larger/smaller width implies precision over/underestimation. Bugs in implementation will cause the distribution to deviate from normality. Some may be difficult to discover in other ways since, for example, may occur only for very rare numerical cases.

To test implementation and basic statistical correctness, the signal is generated according to the model assumed by the basic method (Eq. 3.3). 5000 signals were generated and for each of them the average frequency was estimated with the basic method. Distribution of resulting values of Δ is shown in Fig. 3.4. The fact that it's normal ensures that in this case parameters are estimated correctly.

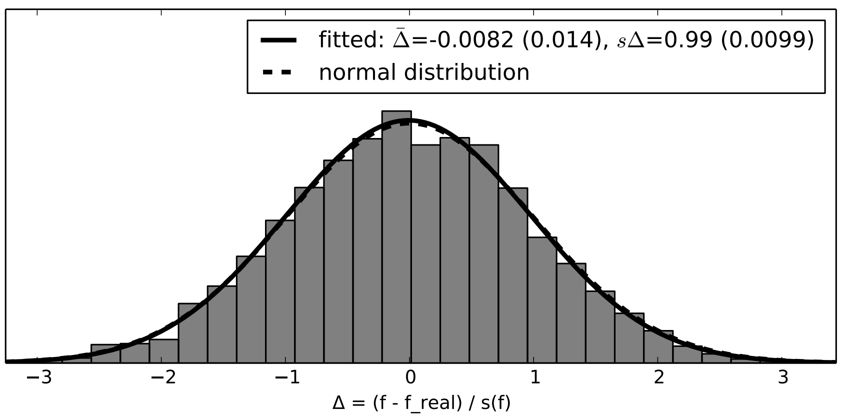


FIGURE 3.4: *Distribution of* $\Delta = \left(E\left(\overline{f}\right) - \overline{f}\right)/s\left(E\left(\overline{f}\right)\right)$ *over 5000 signals, as given by the basic method.*

3.7 Accuracy and precision estimation in presence of frequency drifts

It is clear that both precision and accuracy depend on various experimental conditions, most notably **cycle length**, **magnetic field drifts magnitude**, **signal to noise ratio** and **relaxation time of the signal**. Their influence will be discussed more in detail. Now, for sake of this example, the values corresponding to the optimal functioning of the comagnetometer were chosen:

cycle length is set to 180 s

magnetic field drifts are random 3rd degree polynomials. They model drifts in the order of 10^{-4} Hz for cycle length. Examples are plotted in Fig. 3.5.

relaxation times of the signal are $\tau_1 = 10\,\mathrm{s}$, $\tau_2 = 120\,\mathrm{s}$

signal to noise ratio is $\mathrm{SNR} = 1200$

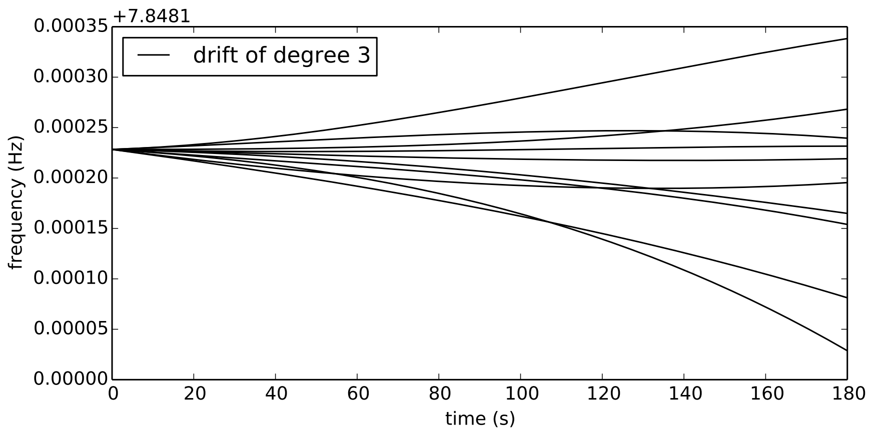


FIGURE 3.5: *Several example frequency drifts modelled as 3rd degree polynomials. Magnitude is normally distributed around 0 (no drift) with width* 10^{-4} *Hz.*

In order to assess precision and accuracy of average frequency estimation with use of a given *method*, a distribution of *errors* is investigated. An *error* is the difference between the estimated and the true, analytically calculated value of the average frequency. In the distribution of *errors*, the average value corresponds to accuracy and the standard deviation to precision.

Non-zero average *error* may come up only if the average frequency is consequently over- or underestimated. However, there might be effects that act one way for some signals and another for others. Consider the following example: hypothetically, there might be a systematic effect correlated with the sign of the frequency drift. The average frequency might be, say, overestimated if the frequency drops, and underestimated if it rises. If now a distribution of *errors* would be made over a sample of signals with random frequency drifts, the systematic effect would average out and would only be seen as deterioration in precision. To really separate this systematic effect, one has to consider signals with each direction of the drift separately.

There indeed is a well-motivated suspicion that there are systematic effects correlated with frequency drifts. Therefore care was taken to separate them. A diagram presenting algorithm for estimation of precision and accuracy is presented in Fig. 3.6.

The algorithm was applied to *the basic method*. Two histograms: of accuracy and precision, each over 200 random drifts (times 50 signals for each drift), were made. They are shown in Fig. 3.7. A very good way to present these results is an **accuracy–precision plane**, where they are depicted by a point – see Fig. 3.8. Additionally, *Cramer-Rao*

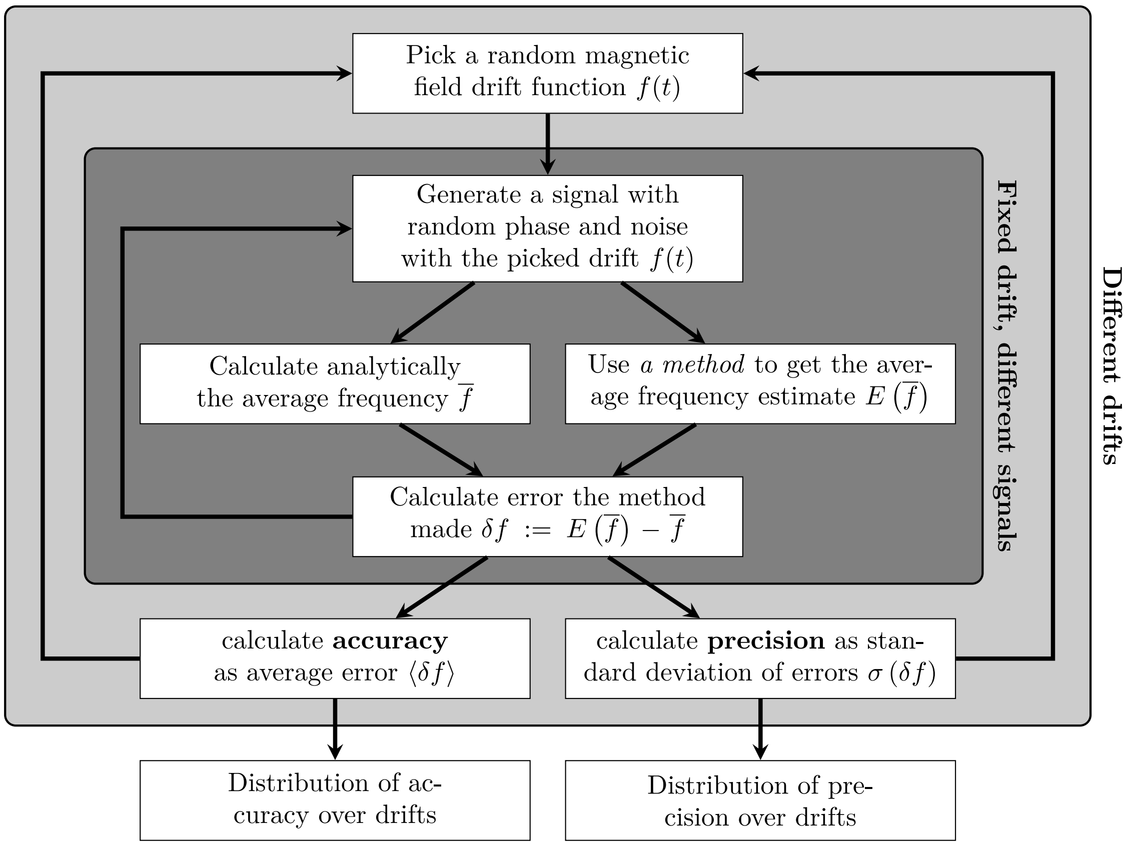

Pick a random magnetic field drift function $f(t)$
Generate a signal with random phase and noise with the picked drift $f(t)$
Calculate analytically the average frequency $\overline{f}$
Use *a method* to get the average frequency estimate $E\left(\overline{f}\right)$
Calculate error the method made $\delta f := E\left(\overline{f}\right) - \overline{f}$
Fixed drift, different signals
Different drifts
calculate **accuracy** as average error $\langle \delta f \rangle$
calculate **precision** as standard deviation of errors $\sigma\left(\delta f\right)$
Distribution of accuracy over drifts
Distribution of precision over drifts

FIGURE 3.6: *Diagram showing methodology of assessing precision and accuracy of average frequency estimation. Care is taken to separate possible systematic effects correlated to magnetic field drifts.*

bound is marked there, which is a theoretical upper limit on the precision, formulated in information theory. For the assumed experimental conditions, the precision of the basic method is estimated to be $\approx \mathbf{10^{-7}\,Hz}$, which is very close to theoretical limit, (*Cramer–Rao bound*), $\approx 5 \times 10^{-8}$ Hz. This means that the average frequency cannot be estimated much more precise than how it is by the basic method. This is intuitive, as in no–drift limit the method would be theoretically an estimator with the least variance. However, accuracy turns out to be $\approx \mathbf{10^{-5}\,Hz}$, that is 100 times larger than precision. This means that value of average frequency estimated by the basic method is actually dominated by systematic effect due to frequency drifts.

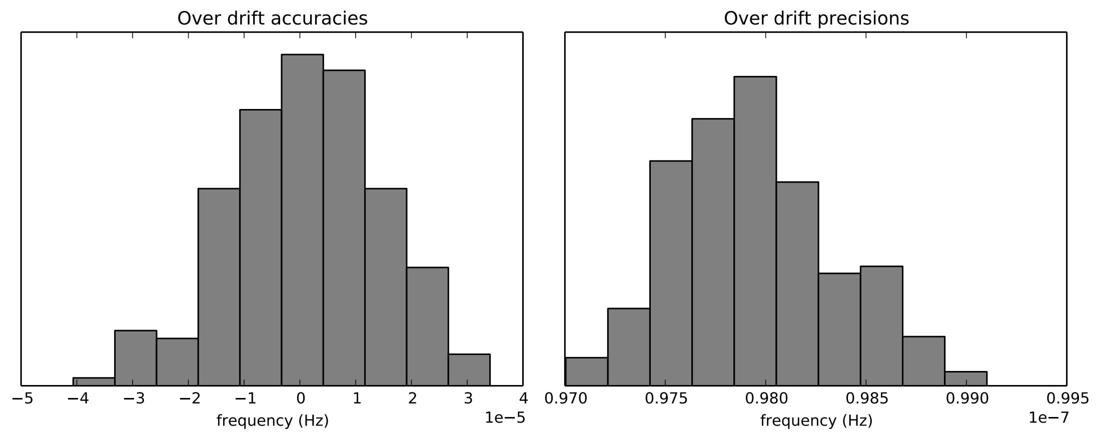


FIGURE 3.7: *Distributions of accuracy (left) and precision (right), each estimated on 50 random signals, over 200 random drifts for the* basic method. *It follows that the method has (in case of the experimental conditions defined in Sec. 3.7),* ***precision*** $\approx \mathbf{10^{-7}}$ ***Hz*** *and* ***accuracy*** $\approx \mathbf{10^{-5}}$ ***Hz****.*

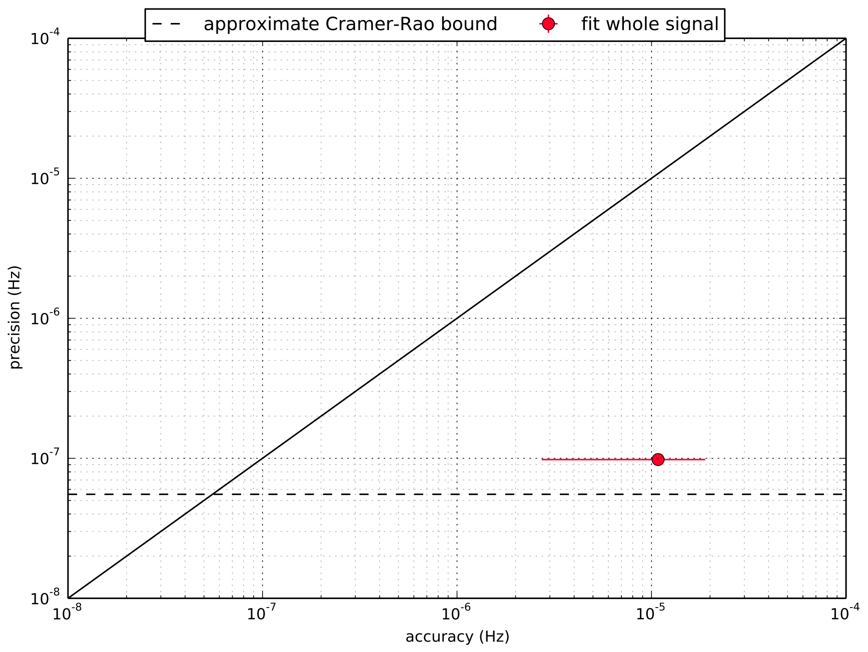


FIGURE 3.8: *Estimated precision and accuracy of the* basic method *of average frequency estimation. The red point represents centres of the precision and accuracy distributions shown in Fig. 3.7. Diagonal black line depicts equality of both parameters to visualise which of the two is dominating – point to its right corresponds to systematic effects larger than the precision. Also* approximate Cramer–Rao bound *(calculated analytically in zero-drift limit) is shown.*

3.8 The filter

In the experimental set-up, the signal coming from the ^{199}Hg comagnetometer is filtered with an analogue hardware band–pass filter. It was shown by Perkowski [4] that it is very well reproduced by a **digital Butterworth band–pass filter** of order 1, with the cut-off frequencies 7.125 Hz and 8.68 Hz. Agreement between the analogue filer and its digital equivalent is presented in Fig. 3.9.

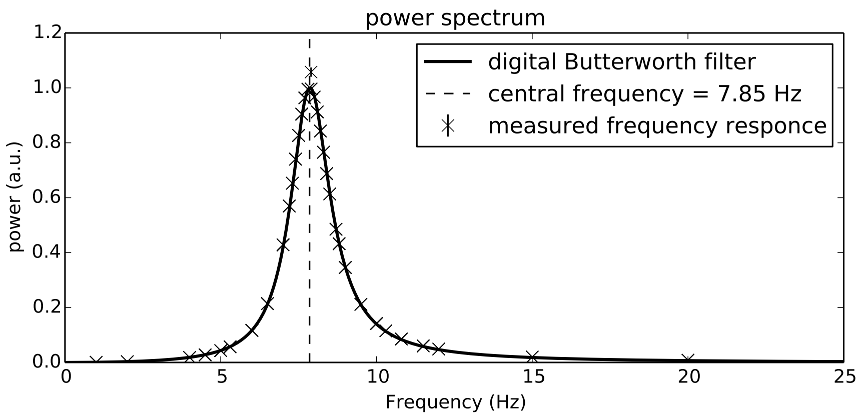


FIGURE 3.9: *The frequency response of the implemented Butterworth filter with central frequency 7.852 Hz and cut-off frequencies 7.125 and 8.68 Hz. Also shown is the frequency response of the hardware analogue filter used in the experimental set-up, measured by Perkowski [4].*

When analysing a filtered signal, it is crucial to understand how the filter changes the filtered signal. Figure 3.10 presents three power spectra: of an unfiltered signal, of a filtered signal and of the transmission of the band–pass filter used in the set-up. The power spectrum of the unfiltered signal has two parts. The first is a sinusoidal signal of interest having a form of a narrow peak (green in the figure). The second part is a white noise, whose power spectrum is uniformly distributed among all frequencies (red in the figure). In particular, the noise has also some power in the band of the interesting signal. Now a band–pass filter, tuned to frequency of the interesting signal, is applied. It will damp most of the noise outside the band of the interesting signal. However, the part inside the band, together with signal itself, will not be touched. Although the overall signal-to-noise ratio is greatly improved, it stays the same in the interesting band [5].

To give a better feeling of this leftover noise, in Fig. 3.11 effects of applying the filter to a white noise are presented. In the frequency domain it is clearly visible that, even though the filter damps almost all the noise, it leaves out the part in its pass–band. Looking at the time domain, it is clear that **on no account** it can be stated, that the noise in the filtered signal has a form of independent, normally distributed random numbers. Rather than that, filtering turns an array of perfectly uncorrelated points into an oscillating, signal–like structure. It is instructive to imagine what happens with an ideal signal when filtered noise, such as described, is added to it. It will be a sum of two sine waves with almost equal frequencies. Therefore, the ideal signal will be distorted in way such, that its phase will fluctuate.

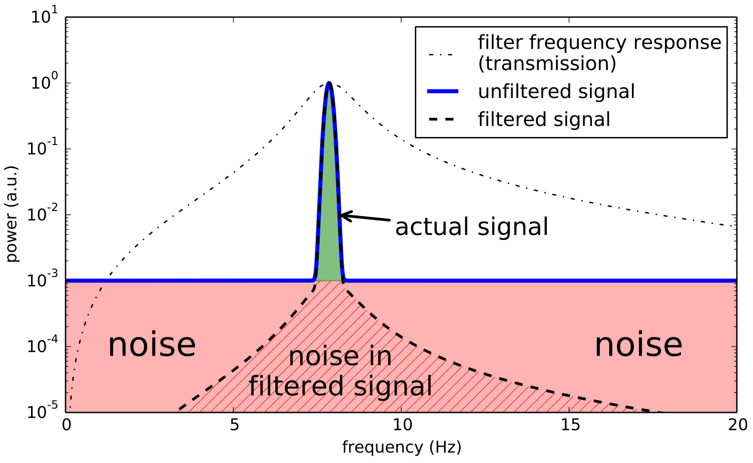


FIGURE 3.10: *Three power spectra: measured signal, transmission of band–pass filter used in set-up and filtered signal.*

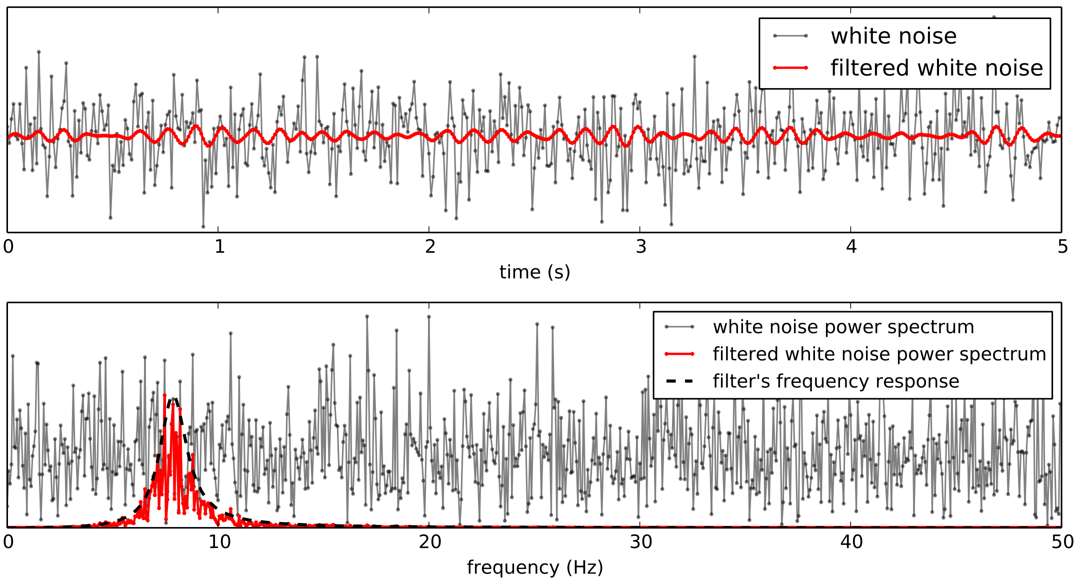


FIGURE 3.11: *Effect of applying band–pass filter to white noise in time (top) and frequency domain (bottom).*

Although filtering is an invaluable tool for signal analysis, it introduces correlations between samples. The greater, the narrower the pass–band of a filter is. Therefore, the presence of a filter in a system, makes a proper, statistically correct, analysis of the signal much harder (but not impossible, as was shown by Chibane et al. [6]). If these correlations are ignored, fitting routines will yield artificially high precision. This is because the filtered data have very little noise indeed, but the original uncertainty still remains. It has another form though — fluctuations of the phase of the signal.

Rather than try to propagate filter–induced correlations, a very pragmatic approach was taken. Recall that the method to estimate precision presented in Sec. 3.7 does not rely on the precision estimate that fitting procedure gives. Therefore, it gives correct results even for filtered signals. The ratio between such obtained correct precision and precision estimates given by fits is called a *correction factor*. It can be used in situations when the only known precision estimate comes from a fitting procedure, as it is with analysis of the real data.

3.9 Two windows method

3.9.1 The original method

It was shown by Chibane et al. [6], that in the case of an ideal signal with a constant amplitude and a slightly variable frequency, the average of the latter is best estimated by the difference between the phases of the signal at its ends:

$$\hat{f} = \frac{\phi_2 - \phi_1}{2\pi T}\,, \tag{3.8}$$

where T is the duration of the measurement, and phases are *cumulative*, i.e. monotonically rising with time, rather than taken modulo 2π. The authors also advised that the phases ϕ_1 and ϕ_2 may be estimated by choosing short fragments of the signal at its beginning and its end. Length of these fragments should be such, that frequency can be assumed to be constant therein.

Green et al. [7] adopted this proposal by defining an algorithm presented in Algorithm 3.1. This routine has been used throughout the nEDM collaboration for six years now. In particular, it is implemented in the data acquisition software to perform the on-line analysis. Results of this algorithm were already used in publications, e.g. by Zenner [8].

Below, a thorough, step-by-step analysis of the, so called, *two windows method* is presented, together with a conclusion.

Algorithm 3.1: Algorithm of signal analysis proposed by Green et al. [7].

```
Input: signal — array of evenly sampled photomultiplier voltage $y_i$, with already
       subtracted DC
Output: average frequency estimator $f$

discard points at the end of the signal until signal–to–noise ratio is $\geq 5$ everywhere
$S_1, S_2 \leftarrow$ two windows with the first and last 15 s of the remaining part of the signal
$T' \leftarrow$ duration of the remaining signal minus 15 s
$N \leftarrow$ count zero-crossings between starts of windows $S_1$ and $S_2$
                    *The next three steps find initial estimates for frequency $f$ and relax-
                    ation time $\tau$.*
$f \leftarrow \frac{N}{T'}$
$R \leftarrow \ln\left(\frac{\text{average } y_i \text{ in } S_1}{\text{average } y_i \text{ in } S_2}\right)$
$\tau \leftarrow \frac{T'}{R}$
for *two iterations* do
    *The first iteration gives improved initial estimates of $f$ and $\tau$, which
    are then used in the second one.*
    for $j \leftarrow 1, 2$ do  *(both windows)*
        $a_j, b_j \leftarrow$ linear least–squares fit with model:
            $y = e^{-t/\tau}\Big(a_j \sin\left(2\pi f(t - t_j)\right) + b_j \cos\left(2\pi f(t - t_j)\right)\Big)$
            $f$ and $\tau$ are fixed in the fit. $t_j$ is the beginning of the $j$th window.
        $A_j \leftarrow \sqrt{a_j^2 + b_j^2}$
        $\phi_j \leftarrow \operatorname{atan} \frac{b_j}{a_j}$
    end
    $\tau \leftarrow \frac{T'}{\ln(A_1/A_2)}$
    $f \leftarrow \frac{\phi_2 - \phi_1 + 2\pi N}{2\pi T'}$
end
```

3.9.2 Implementation

Keeping in mind that the method is intrinsically iterative, one might wonder whether it is certain that it converges sufficiently after just two iterations. In fact, Zejma [9] found that there are cases, where only two iterations are not sufficient. Therefore, to remove the need of iteration, a routine performing a non-linear least–squares fit to an oscillating signal was implemented, presented as Algorithm 3.2. To keep maximal generality, the algorithm is designed to be able to fit sinusoidal signals with additional parameters, such as a variable amplitude or a DC component.

The two windows method with the above algorithm replacing the original iterative linear fit was implemented. Also, in the implementation no points were discarded, regardless

Algorithm 3.2: General routine to perform least-squares fit with a sine-like function.

```
Input: signal $y_i$, sampling rate, functional model to be fitted with parameters:
       frequency $f$, phase $\phi$ and set of others $P$)
Output: estimators of model's parameters and their precision estimators

$f_0 \leftarrow$ FFT of the signal
if signal longer than 10 periods then
    $f_0, \phi_0, P_0 \leftarrow$ recursively call the algorithm for subset of the signal containing only
    the first 10 periods, but no more than half of the whole signal
else
    $P_0 \leftarrow$ reasonable initial estimates, eg. $A_0 \leftarrow \frac{1}{2}(\max y_i - \min y_i)$, $DC \leftarrow \overline{y}_i$
    $\phi_0 \leftarrow$ fit model with $f$ fixed to $f_0$ and $P$ to $P_0$ — only $\phi$ parameter is fitted
end
$\hat{f}, \hat{\phi}, \hat{P} \leftarrow$ fit with initial conditions $f_0, \phi_0, P_0$
if reduced $\chi^2$ of fit $> 1000$ then
    try also fit with $\phi_0 \leftarrow \phi_0 + \pi$
end
```

the signal–to–noise ratio. Tests were performed for such implemented *two windows method*, in a way described in Sec. 3.6.2. In results, there were cases with frequency estimator far-off from the real value. The issue was identified to be a delicate problem with counting zero–crossings in the signal. Artificial zero–crossings show up for poor signal–to–noise ratio, which happens for short relaxation times τ (see Fig. 3.12). Just one extra artificial period changes the result by the inverse length of the signal $\frac{1}{180}\,\mathrm{s} \approx 5 \times 10^{-3}\,\mathrm{Hz}$. The problem can be solved by finding zero–crossings, which are separated by less than 0.75 of the period length and removing one in each such pair.

Furthermore, occasionally a problem arises at the boundaries. When a signal starts with phase $\phi \approx 0$, a mistake can be made in two ways (see Fig. 3.13):

1. First data point is negative and the second is positive, but fitted phase is slightly above zero.

2. First data point is already positive, but fitted phase is just below 2π.

In the first case, an extra zero–crossing is found. In the second scenario, a zero–crossing is missed. An analogous problem may occur at the end of the signal. The issue is addressed by detecting those cases and adding or removing crossings appropriately.

With the above problems solved, the two windows method described above passed the *implementation test* (Sec. 3.6.2) giving normally distributed Δ.

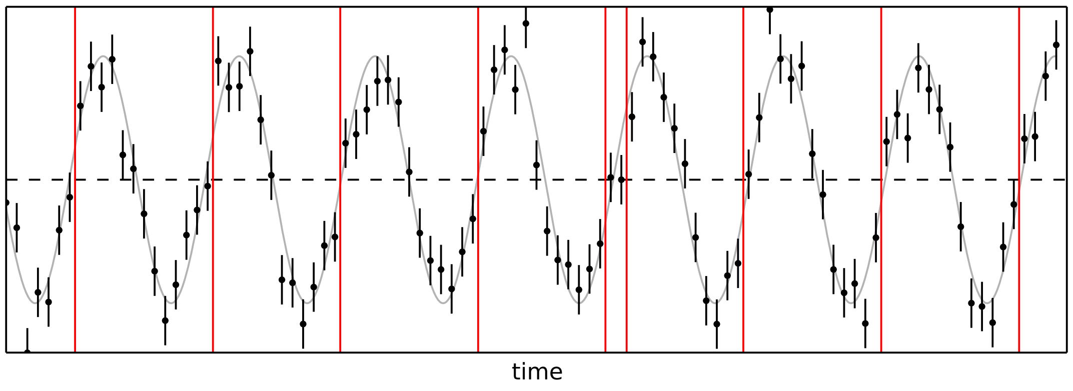


FIGURE 3.12: *Signal with signal-to-noise ratio $\approx$ 5. Red vertical lines indicate where signal changes sign from $-$ to $+$. Notice that in one place sign changes twice for one period.*

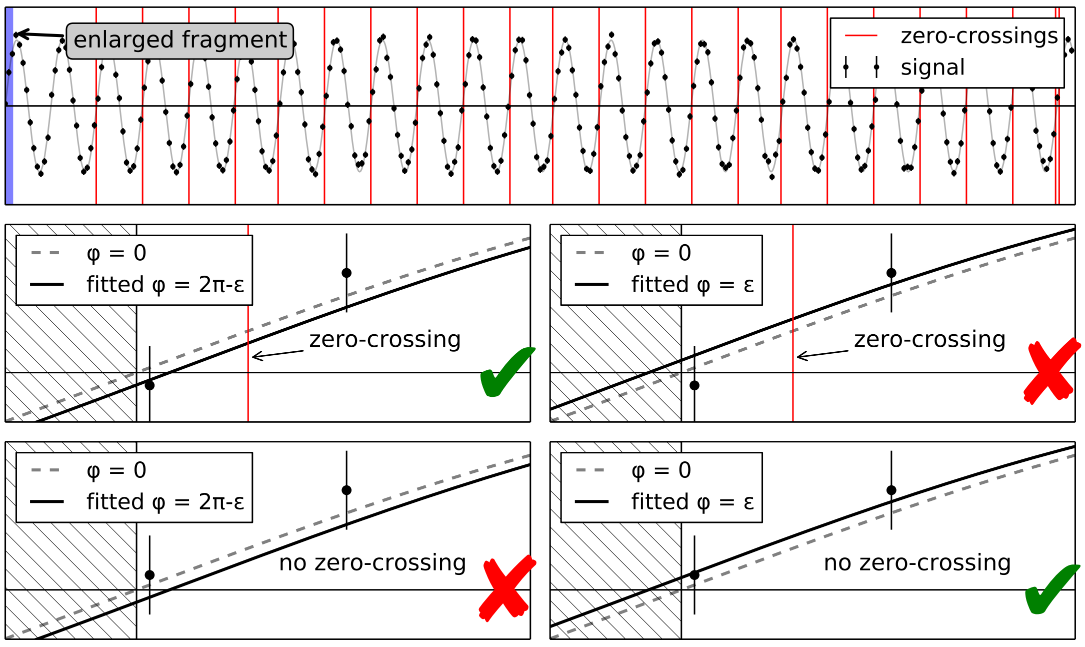


FIGURE 3.13: *Explanation of possible mistake at the beginning of signal when counting periods. If fitted phase at the beginning is $2\pi - \epsilon$ a zero-crossing is expected. Similarly, if fitted phase is small crossing is not expected.*

3.9.3 Location of ϕ_2

Chibane et al. [6] stated, that ϕ_2 should be estimated at the very end of the measurement period. It is of most importance when frequency of a signal fluctuates, which it does when magnetic field drifts. Remembering that we are interested in the average frequency during the whole UCNs storage period, choosing ϕ_2 not at the end will cause systematic effect connected with direction of the drift. For example, if magnitude of the magnetic field rises, the dropped part (after ϕ_2 location) would contain signal with the highest frequency and thus discarding it will underestimate the average.

In the original method ϕ_2 is located at the start of the second window, i.e. $\approx 15\,\mathrm{s}$ before the end. Results of simulation (with an assumption of very good experimental conditions) are presented in Fig. 3.14. They show that fitting phase at the beginning of the second window causes a drift–related systematic effect that may be even several times larger than precision. A proper fit, with the second phase estimated at the end of the second window, gives much better accuracy of the estimated frequency. The method with the ϕ_2 estimated at the end of the signal, can be easily implemented by fitting an inverted data sequence in the second window and taking $\phi_2 = \pi - \phi_2^{inv}$.

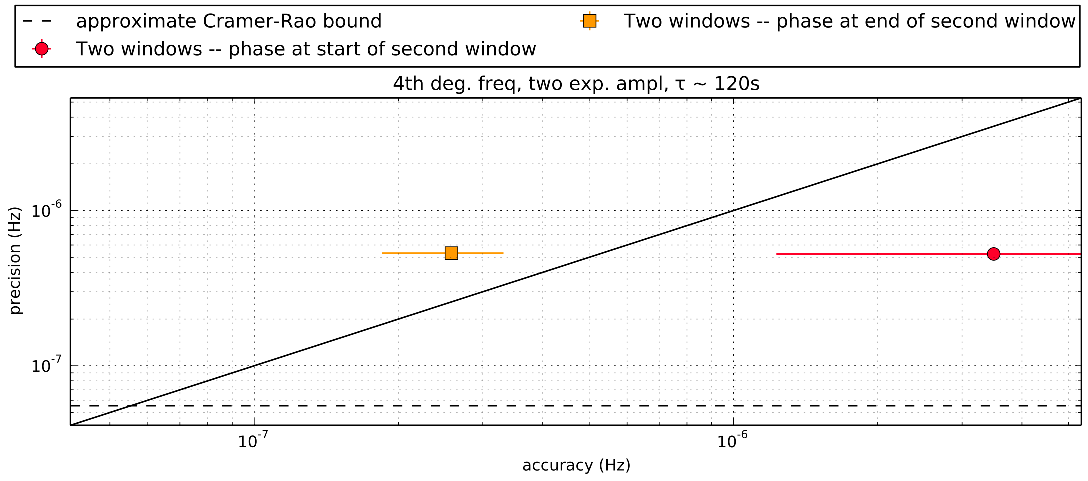


FIGURE 3.14: *Accuracy and precision of two variants of* two windows method*: with phase fitted at the beginning and at the end of the second window. The simulation was carried out with: $\tau = 120\,s$, SNR$=1200$, magnetic field stability $\approx 10\,pT$, filter was included.*

Furthermore, in the original method the authors decided to discard last part of the signal when signal–to–noise ratio there is unsatisfactory. Because of that, in *the original method* loss in signal–to–noise ratio induces **systematic** errors rather than increase **statistical** uncertainty, as it should.

3.9.4 Size of windows

In the original method a very important parameter, or actually two – sizes of two windows, are put forward without any discussion. Intuitively, the larger they are the more precise result will be, but also more susceptible to systematic errors due to the frequency not being constant. A series of simulations were performed, as described in Sec. 3.7, to estimate accuracy and precision of the *two windows* method in function of sizes of the windows. Results are shown in Fig. 3.15. Two important conclusions may be drawn from the plot. First is that making the first window twenty times smaller than the second one hardly influences results. The reason is much bigger uncertainty of the second fit, which is itself caused by a smaller signal–to–noise ratio there. Secondly, the plot allows one to choose the windows size, given a desired upper limit on a possible systematic error, so that precision is maximal. It confirms, that for a good signal relaxation time $\tau = 120\,\mathrm{s}$ and a good signal–to–noise ratio of 1000, windows of length of $15 - 20\,\mathrm{s}$ are a good choice.

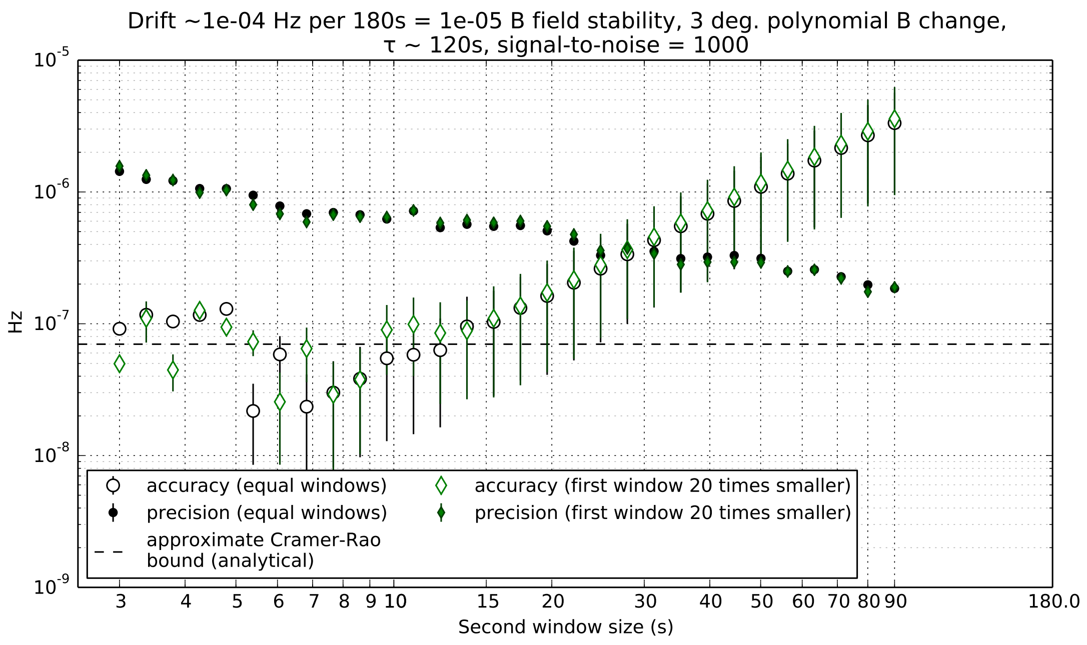


FIGURE 3.15: *Accuracy and precision of the two windows method versus size of the second window. Assumed signal parameters are listed above the graph. Two cases are presented: with both windows of the same size and with first one being 20 times smaller then the second. Also Cramer-Rao bound, calculated analytically in no-drift limit, is marked.*

It is important to realise, though, that results will vary depending upon experimental conditions measurement is done in, mainly relaxation time of the signal τ, signal-to-noise ratio and magnetic field stability. Figure 3.16 presents the same plot for $\tau = 40\,\mathrm{s}$, revealing that in such case precision in relation to drifts is low enough to allow even $50\,\mathrm{s}$ windows not to cause any significant accuracy loss.

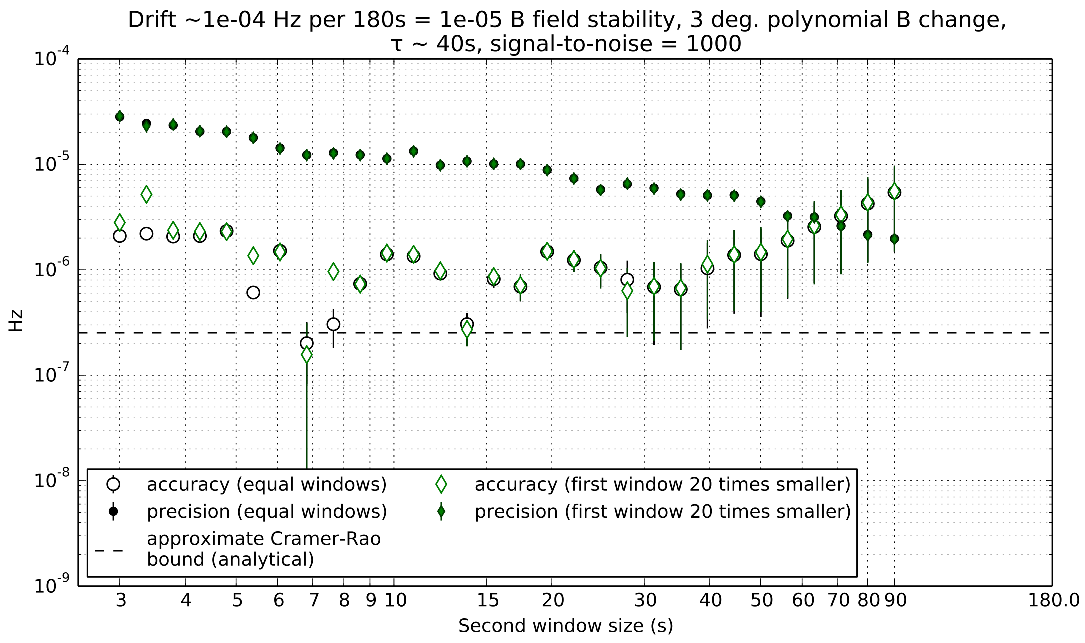


FIGURE 3.16: *As Fig. 3.15, but for much shorter relaxation time $\tau = 40$ s.*

3.9.5 Influence of the filter

Results presented so far presented an analysis of an idealised, not filtered signal. In the experimental set-up, though, signal goes through a band–pass filter (see Sec.3.8) and a possible influence of this fact needs to be investigated.

As it turns out, there are two main effects. First is that fitting routines give overestimated precision estimators in the case of filtered signals, as it was already discussed in Sec. 3.8. Figure 3.17 shows juxtaposition of filtered and not filtered cases. It is clearly visible that the precision estimated by the fitting routines is much lower than the real precision.

The simulations, whose results are shown in Figure 3.17, were carried out with the frequency of the signal being very close to the central frequency of the filter. The other effect manifests itself when these frequencies differ. Figure 3.18 shows how the accuracy deteriorates with increasing mismatch between them. Even 0.1 Hz difference is enough to cause a systematic effect larger than the precision. The precision itself stays constant, so the effect is not visible when the real data are analysed.

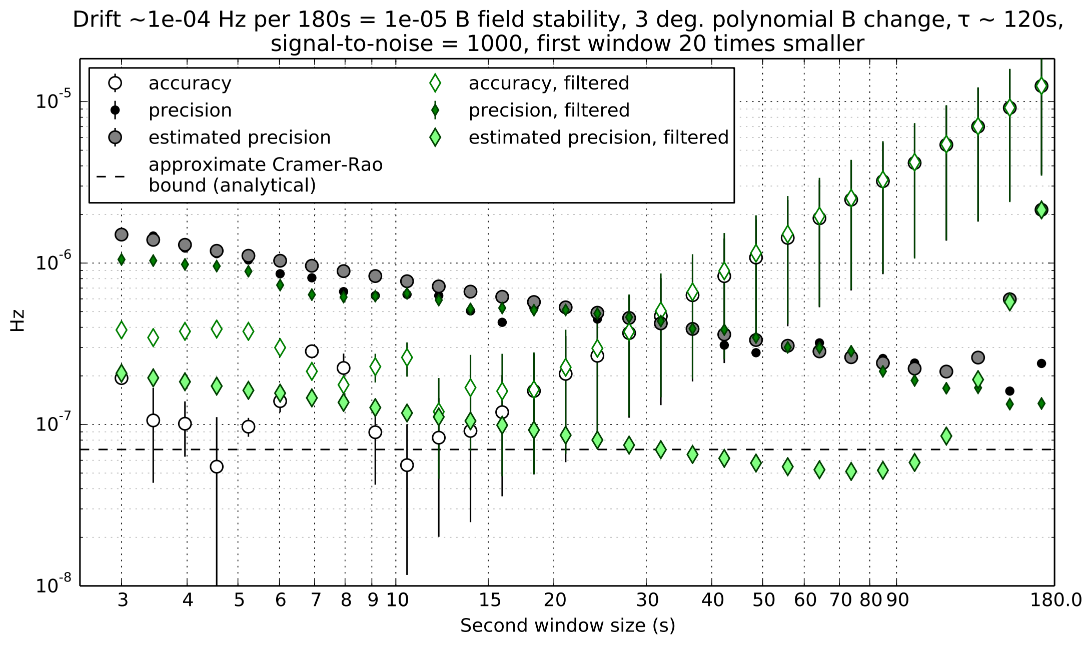


FIGURE 3.17: *Simulation results concerning influence of the filter on the* two windows *method.* Estimated precision *is precision estimator returned by the fitting routines. The frequency of the signal was very close to the central frequency of the filter.*

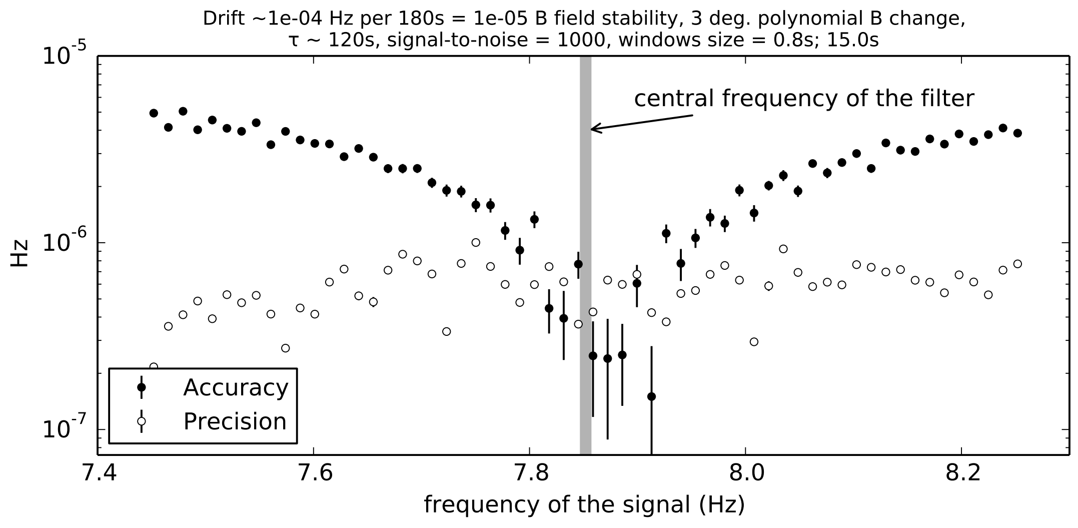


FIGURE 3.18: *Accuracy and precision of the* two windows *method used to analyse a filtered signal, in function of a mismatch between frequency of the signal and central frequency of the filter.*

3.9.6 Conclusion

It must be stressed, that in the two windows method the sizes of the window should be optimised, especially in the off–line analysis. In particular, the first window should be shorter then the second, because of the difference in signal–to–noise ratio at the beginning and at the end of the FID signal. The inequality in length should be taken into account also in the on–line analysis.

3.10 *Point-by-point* phase analysis method

3.10.1 Description

In the already cited paper, Chibane et al. [6] assumed a signal of a constant amplitude. They pointed out, that finding amplitude and DC offset (which decreases number of degrees of freedom by 2) allows one to transform a signal y_i into a *cumulative phase array* θ_i, which is then analysed further.

First of all, their line of thought may easily be generalised into signals with variable amplitude, but more degrees of freedom have to be spent. For example, if amplitude is modelled as a sum of two exponents:

$$A(t) = A_1 e^{-t/\tau_1} + A_2 e^{-t/\tau_2}\,, \tag{3.9}$$

there are 4 parameter which, together with DC, decrease the number of degrees of freedom by 5. This simple model may not always be sufficient because of various experimental effects, which may cause the signal to be distorted (one of which is described in Sec. 3.12). A more sophisticated model was thus introduced: the signal is divided into N parts (t_n, t_{n+1}). In each part amplitude is described by a model:

$$A\left(t, t \in (t_n, t_{n+1})\right) = A_n e^{-t/\tau_n}\,, \tag{3.10}$$

with A_n and τ_n being fitted independently in each part. Therefore there are $2N + 1$ parameters in total, including DC.

Secondly, Chibane et al. [6] did not publish a practical way to make the transformation into the *cumulative phase array.* The problem is not very complicated, but not trivial either. To clarify, a full routine is presented in Algorithm 3.3, illustrated in Fig. 3.19.

The problem of the average frequency estimation may be stated, as finding the difference between value of the *cumulative phase* at the beginning and at the end of the signal. Chibane et al. [6] solved this by fitting a straight line to the *cumulative phase* array and taking its slope, divided by length of the signal, as estimator of the frequency. In fact, they showed that it is a least–variance estimator of frequency, in the case of it

Algorithm 3.3: Transformation of a signal into a *cumulative phase* array.

```
Input: signal — array of evenly sampled photomultiplier voltage $y_i$, each at time $t_i$
Output: cumulative phase array $\theta_i$
DC, $A(t)$ ← DC and amplitude function are found with (a single or multiple)
least–squares fits
        The fits are a mathematical model of the signal, which is used in the
        next step.
$t_j^{\pi}$ ← points where phase of the signal is $\frac{\pi}{2}$ or $\frac{3\pi}{2}$
$y_i^{\text{norm.}}$ ← $(y_i - DC)/A(t_i)$
        The $y_i^{\text{norm.}}$ signal is centred around 0 and has amplitude 1
Discard all points where $|y_i^{\text{norm.}}| > 1$
        It is done, so that an arcsin function may be used. Alternatively,
        their values may by changed to 1 (or -1). These points carry least
        information about phase anyway.
$\phi_i$ ← $\arcsin y_i^{\text{norm.}}$
        The $\phi_i$ array is a triangular wave, centred around 0 and values in the
        range $(-\frac{\pi}{2}, \frac{\pi}{2})$. The values represent phase. The $t_j^{\pi}$ points divide the
        triangle signal $\phi_i$ into parts, where it is linear. In the subsequent parts,
        the slope of the array is alternately positive and negative.
$\phi_i^{+}$ ← Multiply by $(-1)$ points in those parts of $\phi_i$, where slope is negative.
$\theta_i$ ← $\phi_i^{+} + M_i\pi$, where $M_i$ is number of part which $i$th point is in.
        The $\theta_i$ is a phase array, which monotonically increases with time.
```

being constant. The whole process, starting from determination of $A(t)$ function, up to the straight line fit, is illustrated in Fig. 3.19. Unfortunately, the implementation of this method has not passed the *basic statistical correctness test* (Sec. 3.6.2) flawlessly. Figure 3.20 presents distribution of Δ obtained for this method. Results show that precision estimates are overstated by a factor ≈ 1.8. Remember, however, that results in this work take into account *real* precision, as was explained in Sec. 3.7.

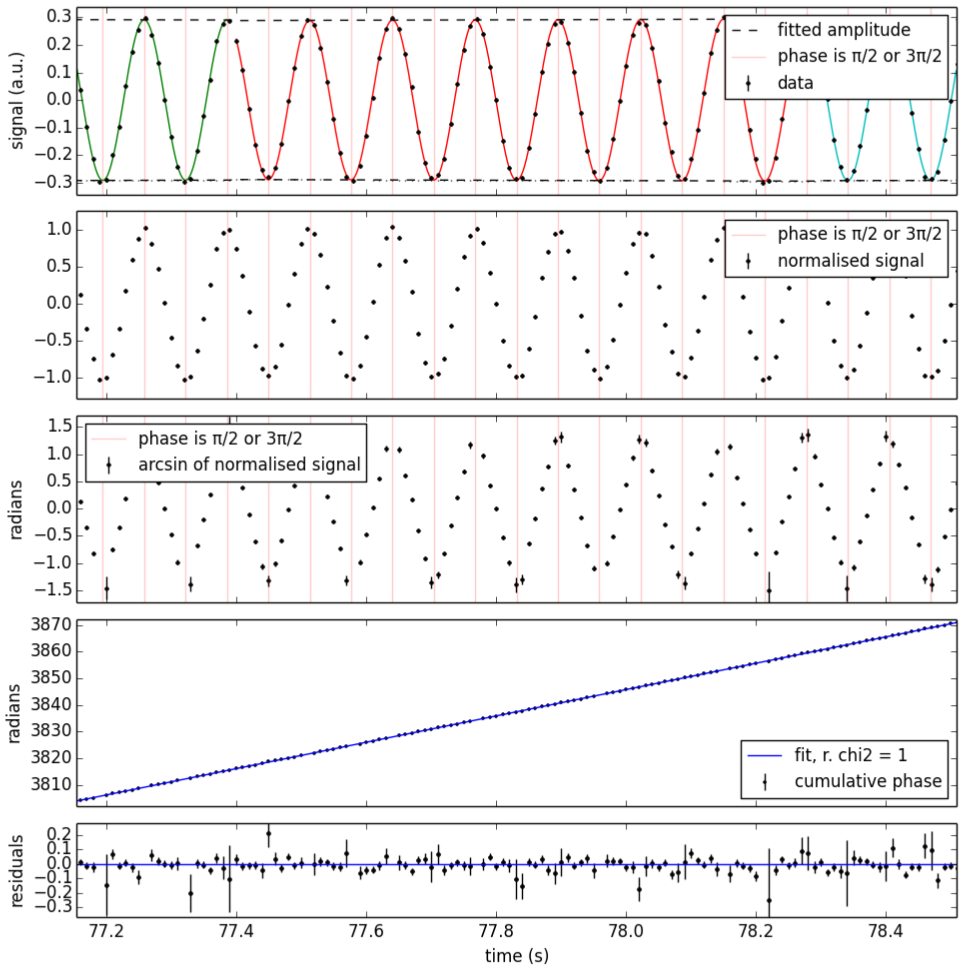
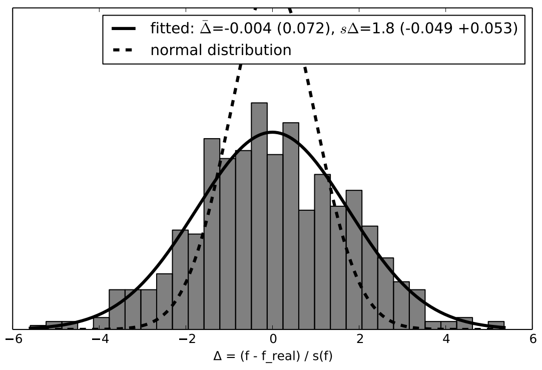


FIGURE 3.19: *Close–up of the point-by-point phase analysis method. From top: 1. Signal is divided into parts and damped sine is fitted in each part. Parts are distinguished by different colours. 2. Amplitude information from fits is used to normalise the signal 3. Arcus sine of normalised signal is calculated. 4. Cumulative phase is derived and straight line fitted to it, whose slope gives frequency estimate. 5. Residuals of the straight line fit.*



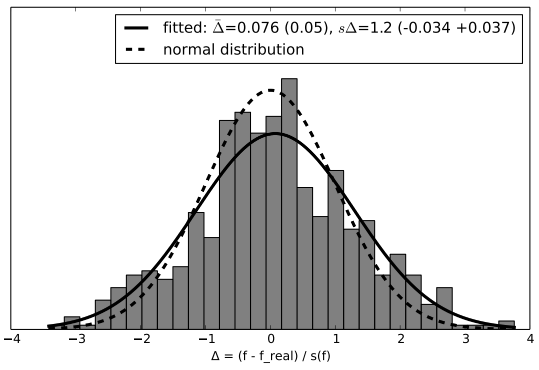


FIGURE 3.20: *Delta distribution in the basic statistical correctness test of the* point-by-point *method with a straight line being fitted to the whole* cumulative phase *array.*

FIGURE 3.21: *Delta distribution in the basic statistical correctness test of the* point-by-point *method with straight lines fitted in two 15 s windows at ends of the signal.*

3.10.2 Case of variable frequency

In case of a variable frequency Chibane et al. [6] advised, that not all points in the signal should be used to estimate the initial and final phases. Rather than that, some first should be used to estimate value of the *cumulative phase* at the beginning, and some last ones to estimate the value at the end. This may be done by fitting a straight line to the first N_{start} points and taking the line's value at start of the array. Analogous procedure may be carried out at the end with N_{end} points. This closely resembles the previously discussed *two windows method.* In fact, when compared, the two procedures give nearly perfectly same results – Fig. 3.22. Results of the *basic statistical correctness test* for the point-by-point version show, that precision is overestimated by a factor of 1.2 (see Fig. 3.21).

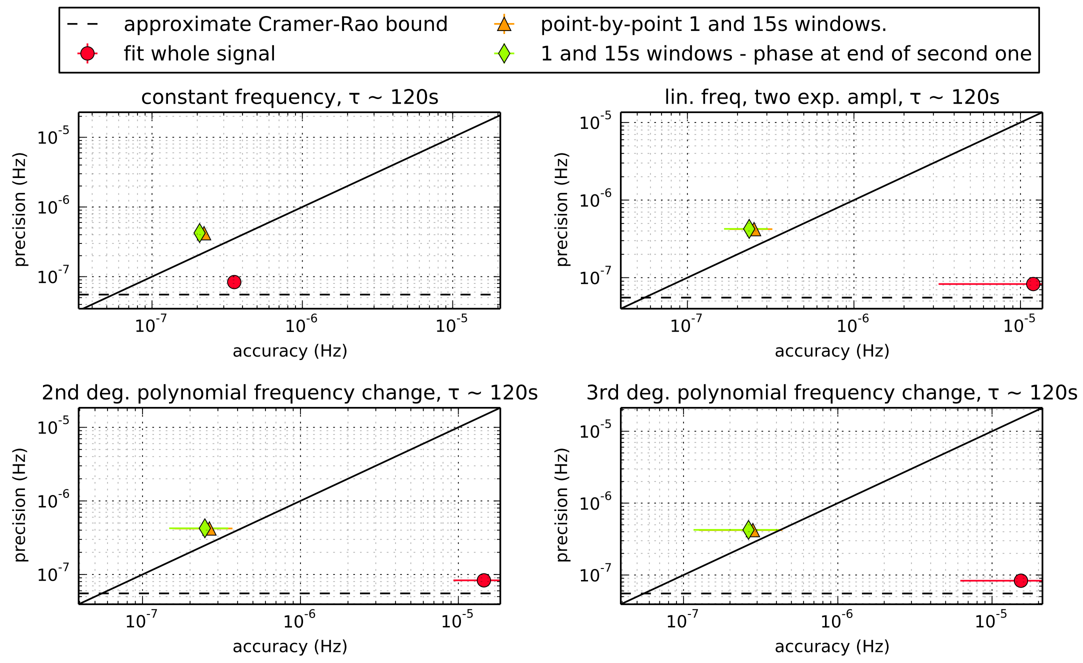


FIGURE 3.22: *Comparison of the two windows method and analogous one based on analysis of the* cumulative phase *array (the* point-by-point *method with the same dimensions of both fitting windows (1 s and 15 s, respectively). Result of the* basic method *(fitting the whole signal) is shown as a reference. Four panels show results obtained for different kinds of frequency drifts.*

There is, however, an another idea on how to handle variable frequency. Namely, one can use all points of the *cumulative phase* array, but fit a polynomial to them. Simulations were performed to investigate how does degree of polynomial fitted to the *cumulative phase* array influences the accuracy and the precision. First set of simulations illustrates an idealised situation: signal has a constant frequency and is not filtered. Results, presented in Fig. 3.23, show that the precision drops with increasing degree of the fitted polynomial. The accuracy is always much better than precision in this case.

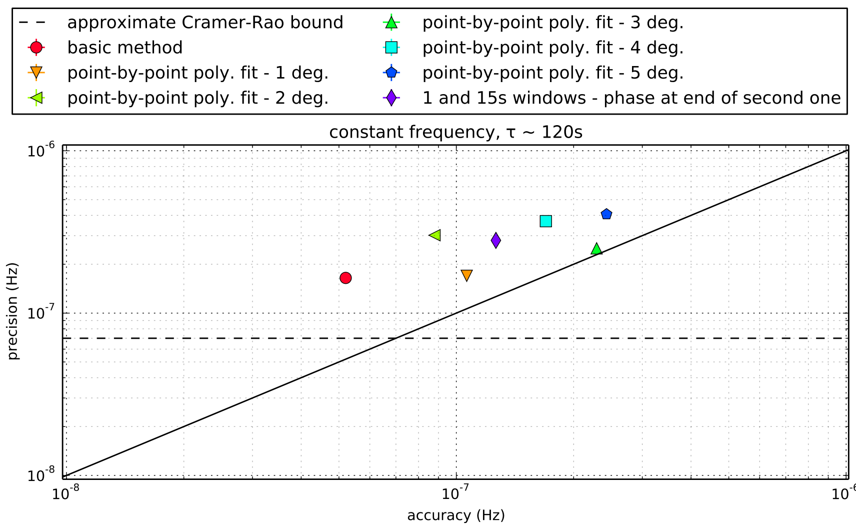


FIGURE 3.23: *Precision and accuracy of* point-by-point phase analysis *method in idealised conditions: constant frequency and signal not being filtered, compared to* the two windows method *and* fitting whole signal.

To assess what might be expected in analysis of the real data, a series of simulations with variable frequency and filtered signal were also carried out, shown in Fig. 3.24. As expected, systematic effects show up when the degree of the fitted polynomial is lower than that of the *cumulative phase* (which is degree of frequency change plus one). The exception is, that even for signals with constant frequency, there is a systematic effect. It disappears when a polynomial of dergee at least 2 is fitted to the *cumulative phase* array. This may be interpreted as an artificial linear drift in frequency induced by the filter.

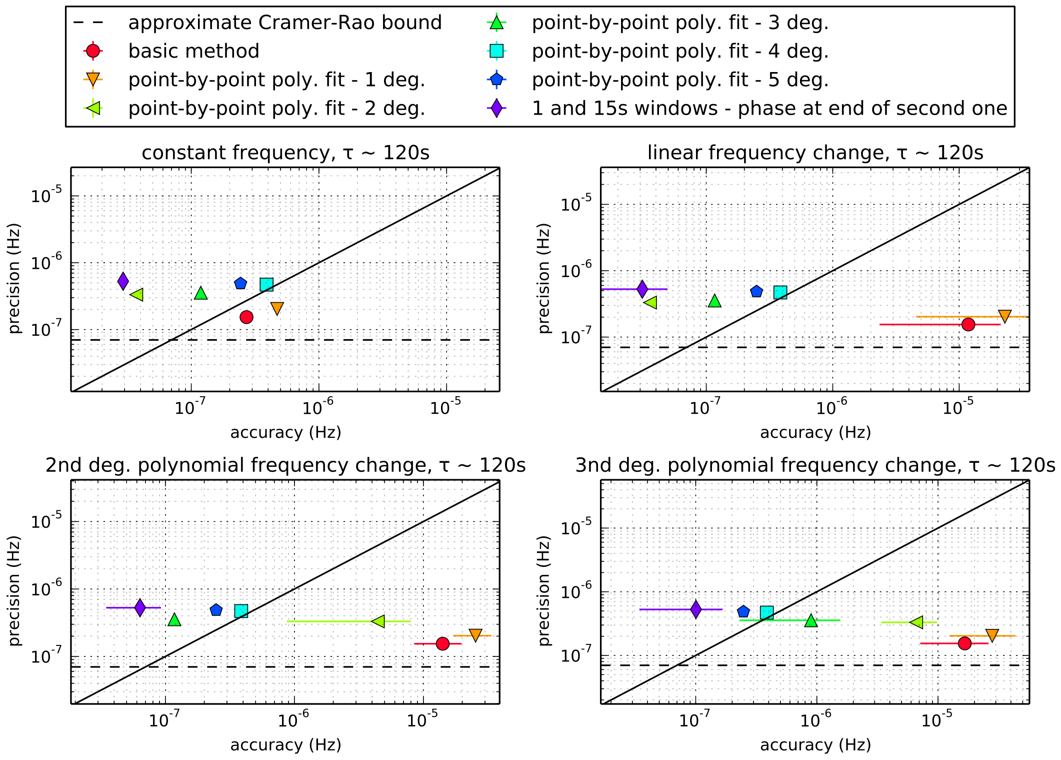


FIGURE 3.24: *Performance of* point-by-point phase analysis *method in more realistic conditions: variable frequency and signal being filtered, compared to* the two windows method *and* fitting whole signal.

3.10.3 Summary

Chibane et al. [6] at the very beginning proposed that analysis of the signal should be based on the concept of *cumulative phase*. They showed that this leads to a least–variance estimator. Additionally, as it is shown in the above sections, even more sophisticated methods, such as the *two windows*, can be realised in the scheme of *cumulative phase* analysis. Therefore, I propose that the analysis of comagnetometer signal is divided into two general parts:

1. Transformation of each point of the signal y_i into its corresponding *cumulative phase* θ_i.
2. Analysis of the *cumulative phase* array θ_i

Such division makes the problem much more intuitive, while not restraining achievable precision or accuracy. Additionally, the second step encapsulates a more general issue of a time–series analysis.

Two ways to incorporate the variability of the frequency were tested. Fitting a polynomial to the *cumulative phase* array showed to be worse in both accuracy and precision then the *two windows* scheme.

3.11 Windowed phase analysis method

3.11.1 Motivation

When a signal is filtered, individual points do not have normally distributed independent errors, but are correlated to one another instead. Chibane et al. [6] showed, that a least variance frequency estimator for an oscillating signal may be obtained by linear fit to the *cumulative phase array*. However, if the array is constructed as described in the *point-by-point* phase analysis method (Sec. 3.10), it will also suffer from cross–correlations between points. These correlations make statistical analysis, like meaningful χ^2 test, very difficult. The *windowed phase* analysis method focuses on obtaining *the cumulative phase array* with each point having an independent error.

3.11.2 Description

The method involves a division of the signal into N equal parts, called *windows*. N is chosen such, that a size of a single *window* is at least equal to the period of the signal. Then, least–squares fits are performed in each *window* to get phase information. Finally, the information about phase is used to construct the *cumulative phase array*.

Let's first focus on the fits. In general, model function fitted in each window has a form:

$$y(t) = A(t) \sin \left(2\pi f(t - t_0) + \phi\right), \tag{3.11}$$

where a good choice for $A(t)$ is $Ae^{-t/\tau}$. Note, that ϕ is an estimator of phase at time t_0. Rife and Boorstyn [10] showed that variance of ϕ depends on t_0. A particular choice is t_0^*, which minimizes $\operatorname{var}\phi$. In the case of an evenly sampled signal with a constant amplitude, t_0^* lies in the middle of the fitted signal. However, in the case of a variable amplitude, it may be shifted. In practice, the problem of finding t_0^* may be solved numerically. In order to do that, define a function performing a least–squares fit and returning the estimator of $\operatorname{var}\phi$. The function should accept t_0 as an argument. (One–dimensional minimization algorithm proposed by Brent [11] converges for this function in about 30 iterations.) It follows, that phase information given by the fits will have a form of an array of $\phi_n(t_{0,n}^*)$ values, where n numbers the windows. There are N such values and they are not evenly distributed in time.

The next step is to construct the *cumulative phase array*. To do that, the *counting periods* technique may be used. In Sec. 3.9.2 it is explained how it is applied to determine the *cumulative phase* between two points. Here it has to be applied several times, between successive $(t_{0,n},\ t_{0,n+1})$ pairs.

To sum up, the *windowed phase* analysis method involves the following procedure:

1. Signal is divided into N equal parts, each longer than signal period. Frequency is assumed to be constant in each part.
2. In each nth part, the phase $\phi_n(t^*_{0,n})$ is found by a least-squares fit, with $t^*_{0,n}$ chosen such that $\mathrm{var}\,\phi_n$ is minimal. The model used in the fits is:

   $$y = A_n e^{-t/\tau_n} \sin\left(2\pi f_n(t - t^*_{0,n}) + \phi_n\right),$$

   with four free parameters A_n, τ_n, f_n and ϕ_n.
3. The counting zero-crossing technique is used for time spans $(t_{0,n},\ t_{0,n+1})$ to find *cumulative* phases $\phi^C_n(t^*_{0,n})$.

The result is an array of *cumulative phase* of size N, which can be further analysed as described in Sec. 3.10.

3.11.3 Adaptive linear extrapolation

The distinctive feature of the *cumulative phase array* obtained with this method is that its points are not correlated with each other, even when the signal is filtered. This makes a statistical analysis, such as a χ^2 test, straightforward. Recall that the *two windows* method, when applied to a *cumulative phase array*, consists of fitting straight lines in two windows: at the beginning and the end of the array. The sizes of the two windows are free parameters which are predetermined. However, they could be adaptively adjusted using χ^2 test, as proposed by Bison and Heil [12], with the Algorithm 3.4.

Algorithm 3.4: *Adaptive linear extrapolation method.*

```
$N \leftarrow 4$
repeat
    fit line to the first N points of $\phi^i_C(t^i_0)$ sequence
    $N \leftarrow N + 1$
until reduced $\chi^2$ < reduced $\chi^2$ limit
use fitted line to extrapolate phase to $t = 0$
```

Analogous procedure is carried out at the end to extrapolate phase to $t = t_{end}$. Figures 3.25 and 3.26 explain the process graphically. In the case of filtered data, one has to take into account that $\phi^C_n(t^*_{0,n})$ values have underestimated variances estimators. They have to be multiplied by appropriate *correction factor* (Sec. 3.8). In the case of the filter used now in the nEDM experiment this factor is 6.32.

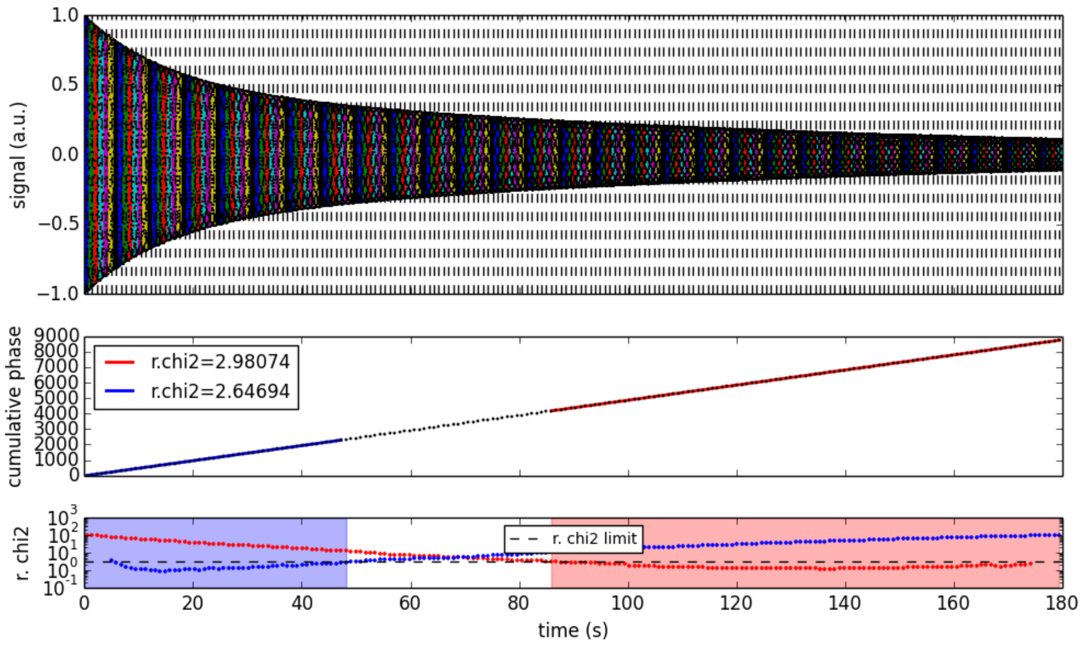


FIGURE 3.25: *Adaptive linear extrapolation in the* windowed phase *analysis method. Data are simulated without filter. From top: 1. Signal divided into 200 parts with vertical dashed lines depicting $t^*_{0,n}$ – points where the phase is determined by fitting the signal inside the given part; 2. Cumulative phase $\phi^i_C(t^i_0)$ sequence with fitted lines (blue and red); 3. Reduced χ^2 of the fit versus location of the end/start of the first/second window. Close-up is presented in Fig. 3.26*

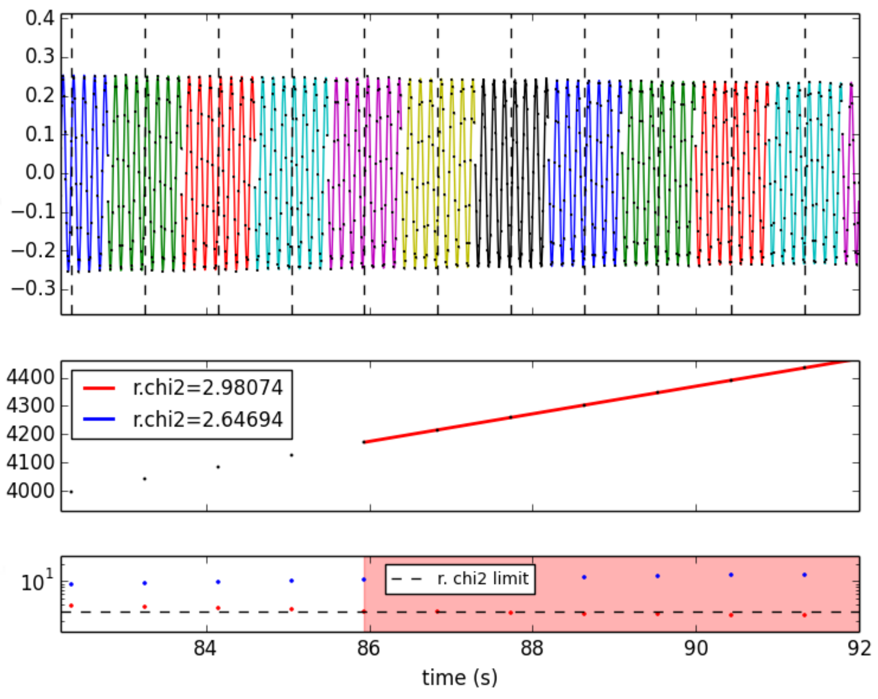


FIGURE 3.26: *Close–up of plot in Fig. 3.25 — only 10 s are shown.*

3.11.4 Performance

If the method is to be used in practice, then reduced χ^2 of the fit in a given window, as a function of the size of this window, must have a proper behaviour. Namely, in the case of a constant signal frequency, the reduced χ^2 is expected to be ≈ 1, regardless the size of the window. In turn, with presence of frequency drifts, it is expected to be ca. 1 for small windows and increase together with the size of the window. It should also not depend strongly on number of parts signal is divided into at the beginning of the analysis.

Size of the second window is much more important, as it was shown that altering the length of the first one barely influences the results. Therefore, firstly Fig. 3.27 presents the behaviour of a reduced χ^2 of only the fit in the second window, as a function of size of the window, in several cases. First, take a look at data with signal not being filtered (depicted with dashed lines). Reduced χ^2 starts with being approximately 1, with frequency drift being both present (thick dashed line) and not (thin dashed line). As expected, the two lines part at certain window size (approximately 120 s in this case), marking the growing influence of the frequency drift. However, the rise is the less distinct, the more parts signal has been divided into. For 400 parts it is barely visible. Unfortunately, the χ^2 rise does not happen at optimal size of window, determined with the method described in Sec. 3.9.4.

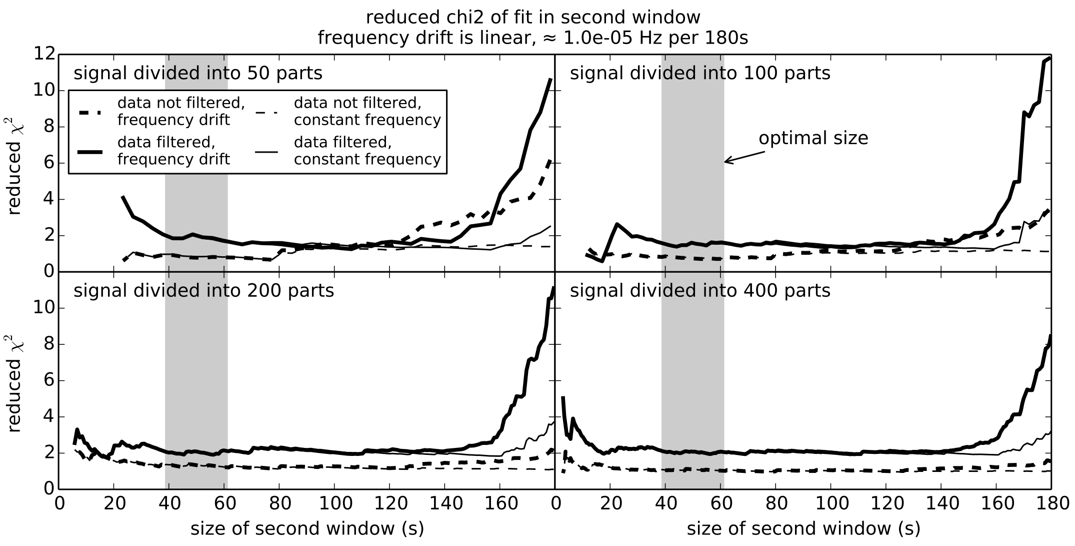


FIGURE 3.27: *Reduced χ^2 of fit in the* ***second window*** *versus the size of the window. Several simulated cases are presented. Thick lines – polynomial drift of the frequency of the signal, approximately 1×10^{-5} Hz per 180 s; thin lines – constant signal frequency; dashed lines – simulation without the filter; solid ones – simulation with the filter. Four panels show results with the signal being divided into 50, 100, 200 and 400 parts. Grey area depicts optimal window size determined as described in Sec. 3.9.4.*

The behaviour changes a little if analysed signal is filtered. Reduced χ^2 at the beginning actually decreases until dropping to ≈ 2. Value of 1 is never reached, even for the signal with a constant frequency, despite the *filter correction factor* having been applied.

Figure 3.28 presents the reduced χ^2 behaviour as a function of the duration of the first of the two windows. The major difference, in comparison to the fit in the second one, is that the reduced χ^2 in a presence of frequency drift begins to rise for much smaller windows — around 50 s, in comparison to ca. 160 s in case of the second window. This shows, that it largely depends on the signal–to–noise ratio, how visible is the influence of the frequency drifts on the signal. In the second window, where SNR is smaller, the drifts hardly have any influence on the χ^2. On the other hand, in the first window, the SNR is large enough for the influence of the drifts to be visible. This is yet another clue supporting the fact, that in the *two windows* type methods of analysis, the first window should be much shorter than the second.

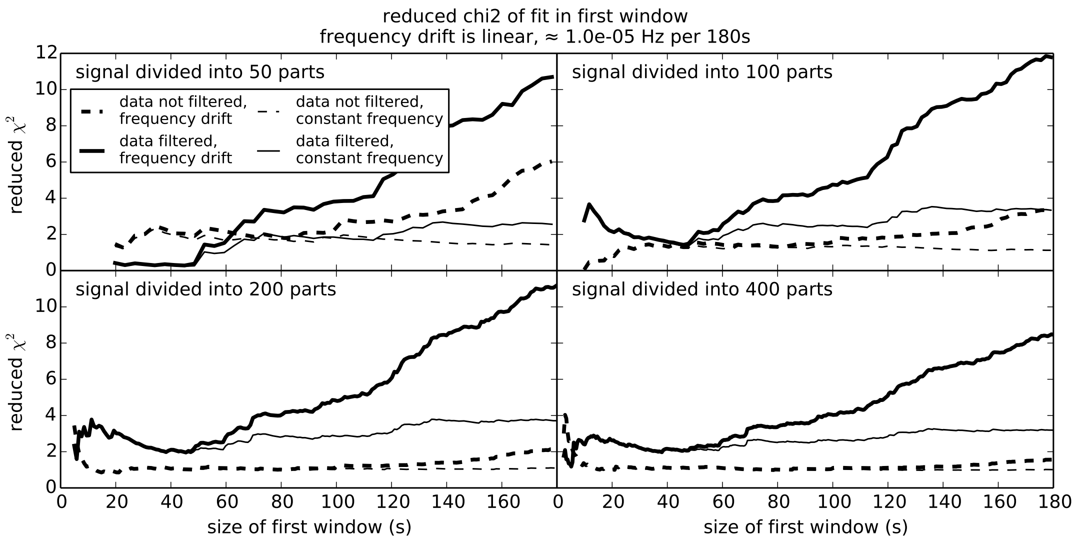


FIGURE 3.28: *As Fig. 3.27, but the duration of the first window is investigated.*

3.11.5 Summary

The method was not investigated further given vague behaviour of reduced χ^2 of fit in function of size of the window. Most importantly, sizes of the windows the method would give are inconsistent with the optimal ones determined with method described in Sec. 3.9.4.

However, it was ruled out that such approach, given enough further inquiry, might give satisfactory results.

3.12 Oscillations of the signal envelope

So far, it was assumed that envelope of the signal of the comagnetometer can be described as a sum of two exponential decays. However, in real data an additional effect was discovered — the envelope of the signal was found to oscillate.

In order to describe it quantitatively, the following procedure was carried out: first, the signal was divided into slices (of size comparable with the period of the signal). Then, sine–wave was fitted to each slice. The amplitudes of fitted waves created an array, which was analysed further. Figure 3.29 shows close-up of the amplitude array, revealing that it oscillates around the sum–of–two–exponents decay.

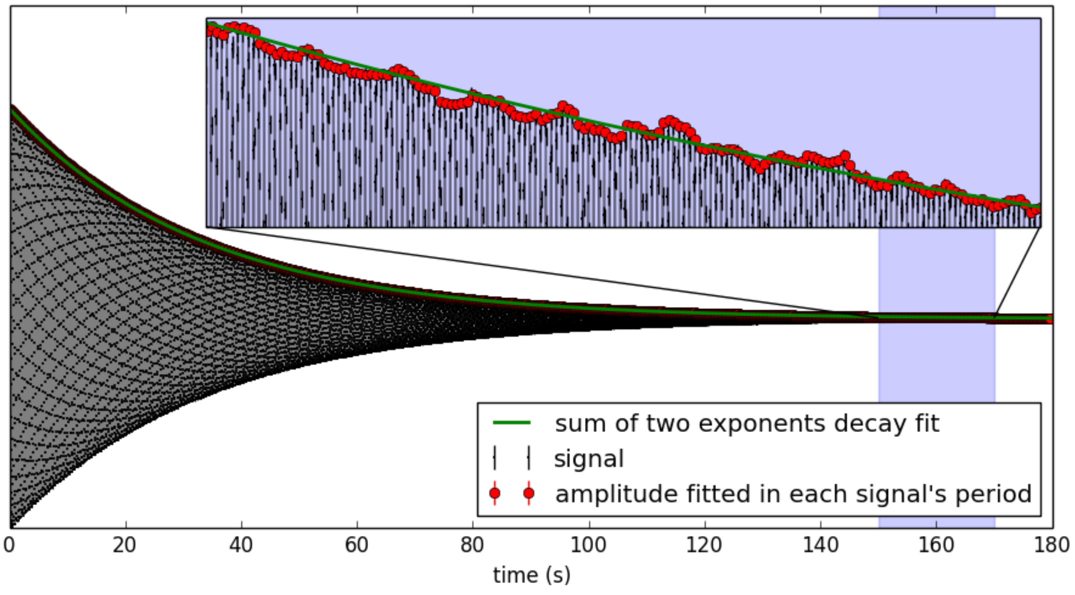


FIGURE 3.29: *The signal measured in the experiment (run 7650). Red dots depict amplitude of sine fitted to each period of the signal, green line — a sum–of–two–exponents fit to the amplitude array. The amplitude is seen to oscillate around the decay line.*

First hundred cycles of run 7650 (done on 16.08.2013) were analysed in such way. To further simplify the problem, only the difference between the amplitude array and idealised sum–of–two–exponents decay was considered. The arrays of differences from the hundred cycles are shown in Fig. 3.30, plotted one on top of another. The plot gives impression that the oscillations have a constant amplitude throughout a cycle and that they do not vary from one cycle to another. It was confirmed, by looking at random individual cycles, that it is indeed true. Figure 3.31 presents power spectrum of the data in Fig. 3.30, revealing that the oscillations have two major frequencies: 2.03 Hz and 3.26 Hz.

Low frequency of the oscillations, and their constancy, might indicate that the source is mechanical (e.g. a vacuum pump). Any beat phenomenon related to signal itself is rather unlikely, since the oscillations retain a constant amplitude even as the signal

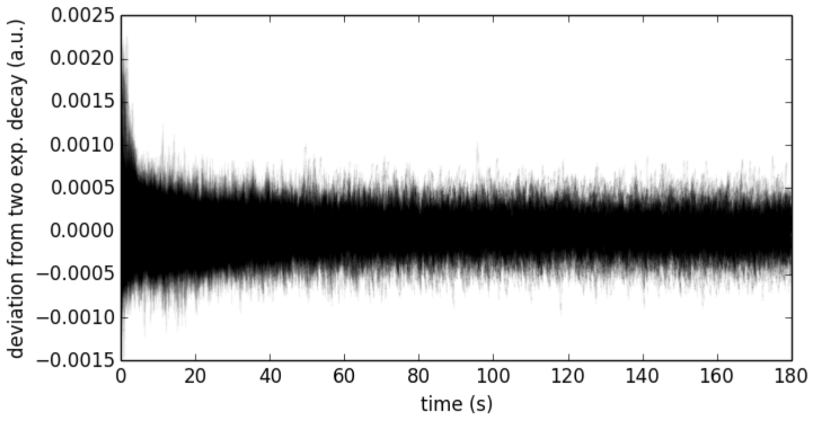


FIGURE 3.30: *Deviation of amplitude of the signal from sum–of–two–exponents decay throughout a cycle. Data from 100 cycles from run 7650 are plotted one on top of another.*

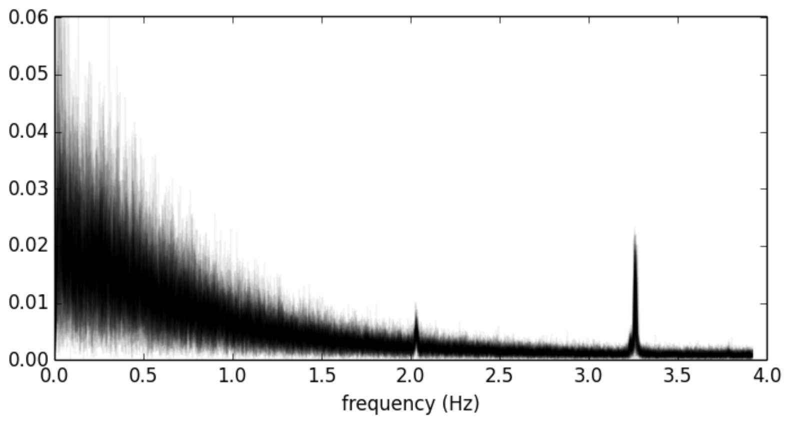


FIGURE 3.31: *Power spectrum of data from Fig.3.30. Two peaks, at **2.03 Hz** and **3.26 Hz**, are clearly visible.*

fades. It is also not an artefact of a beat between sampling and division of signal into slices — no matter how the signal is divided, the phenomenon remains.

It was checked, whether including the oscillations into the simulation has any significant influence on results that the presented methods of analysis give. It turned out, that it has not, justifying negligence of the phenomenon in most of the presented work.

3.13 Data analysis

3.13.1 Proposed scheme

Basing on the results presented above, I propose a following scheme of analysing data of the ^{199}Hg comagnetometer in a given run:

1. Firstly, an initial characterisation of the data must be performed. In order to do that, for each cycle in the run do the following steps:

   (a) Calculate a standard deviation of data taken during the noise measurement (i.e. before ^{199}Hg atoms are filled into the chamber). The result needs to be corrected for the fact, that the recorded noise passed through the filter (see Sec. 3.8). The corrected value σ_{RMS} is an estimate of RMS noise distorting precession data in the following cycle.

   (b) Perform a least–squares fit to the data with the following model:

   $$\left(Ae^{t/\tau_1} + Be^{t/\tau_2}\right)\sin(2\pi f t + \phi)\,.$$

   Algorithm 3.2 may be used to do this.

   (c) Calculate the signal–to–noise ratio using the formula:

   $$\mathrm{SNR} = \frac{A + B}{\sigma_{\mathrm{RMS}}}\,.$$

   (d) Find the first estimate of the average precession frequency $\tilde{\mathbf{f}}$ using the *two windows* method with the windows 2 s and 15 s long (Sec. 3.9).

2. Now the run must be quantitatively characterised, so that later proper simulations may be performed. The run is described by representative parameters: relaxation times of the signal τ_1, τ_2, signal–to–noise ratio and magnitude of the frequency drifts. They are all assessed in the following steps:

   (a) Look at distributions of τ_1 and τ_2 found in the fits done in 1b. Take their averages as representatives.

   (b) Do the same for signal–to–noise ratio.

   (c) Create an array of frequency change from cycle to cycle: $\tilde{f}_{i+1} - \tilde{f}_i$. Usually the drifts span across many cycles (approximately 10) and the cycle–to–cycle difference is dominated by statistical fluctuations. For this reason, smooth the array with a moving average with window size of 10 cycles. Afterwards, take the standard deviation of the smoothed array as representative magnitude of frequency drift.

3. Use the found representative parameters to perform simulations to optimise the sizes of windows in the *two windows* method. In these simulations, set frequency of the signal to approximately central frequency of the filter. The reason is separation of the drift–related systematic effect form the one originating from mismatch between frequency of the signal and central frequency of the filter.

4. Perform simulations to assess how big systematic error is, including the mentioned above mismatch.

5. Analyse each cycle in the run with use of the *two windows* approach realised in the *point–by–point* scheme. Use the found optimal window sizes.

One might argue, that in simulations in step 3 the frequency of the signal should be set as it really is in the analysed run, so that optimisation takes into account the accuracy that might be expected. However, such approach would entangle two effects. First are drift–related effects. Sizes of windows are optimised on account of these. The second one is filter–induced systematic effect, which has to be estimated in step 4.

The above scheme was used to analyse the ^{199}Hg comagnetometer signal collected in two different runs. These example analyses are presented in detail below.

3.13.2 Analysis of run with short τ – run 7650

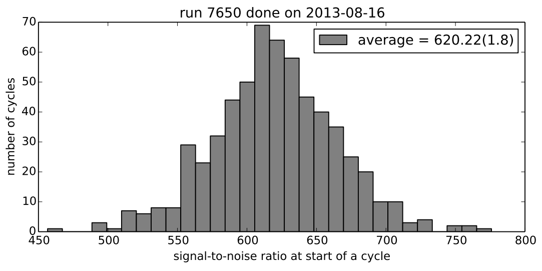


FIGURE 3.32: *Histogram of the comagnetometer signal–to–noise ratio for all cycles of run 7650.*

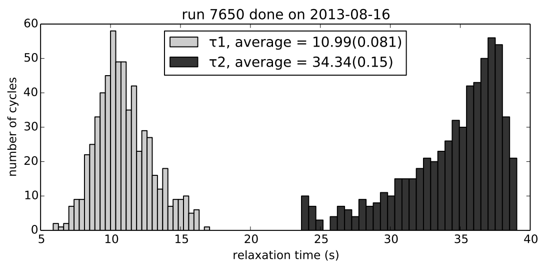


FIGURE 3.33: *Histogram of relaxation times of the comagnetometer signal for all cycles of run 7650.*

In the initial part of the analysis two histograms are created: one of signal–to–noise ratio (Fig. 3.32) and another of relaxation times τ_1 and τ_2 (Fig. 3.33). The histograms serve as a visual check, as for further analysis only their average values are used:

SNR	620
$\boldsymbol{\tau_1}$	11 s
$\boldsymbol{\tau_2}$	34.3 s

The next step, illustrated in Fig. 3.34, is to estimate the magnitude of the precession frequency drifts inside the cycles of the run. In order to do that, each cycle is analysed with the *point-by-point* method with *two windows* (Sec. 3.10.3). The sizes of the windows are initially set to 2 s and 15 s. This gives initial estimators of average precession frequencies for each cycle. Then, a cycle-to-cycle frequency change is calculated. In order to average out statistical fluctuations, the resulting array is smoothed with a *moving average* with window size of 10 cycles. The standard deviation of the smoothed array, in this case $\mathbf{5.4 \times 10^{-6}}$ **Hz**, is an estimate of the magnitude of the cycle-to-cycle frequency drift. Because the drifts have larger time scale than period of execution of the cycles, the value may also serve as an estimate of drift occurring during a cycle. Additionally, an average precession frequency in the whole run: **7.8484 Hz** is found — it is going to be used later.

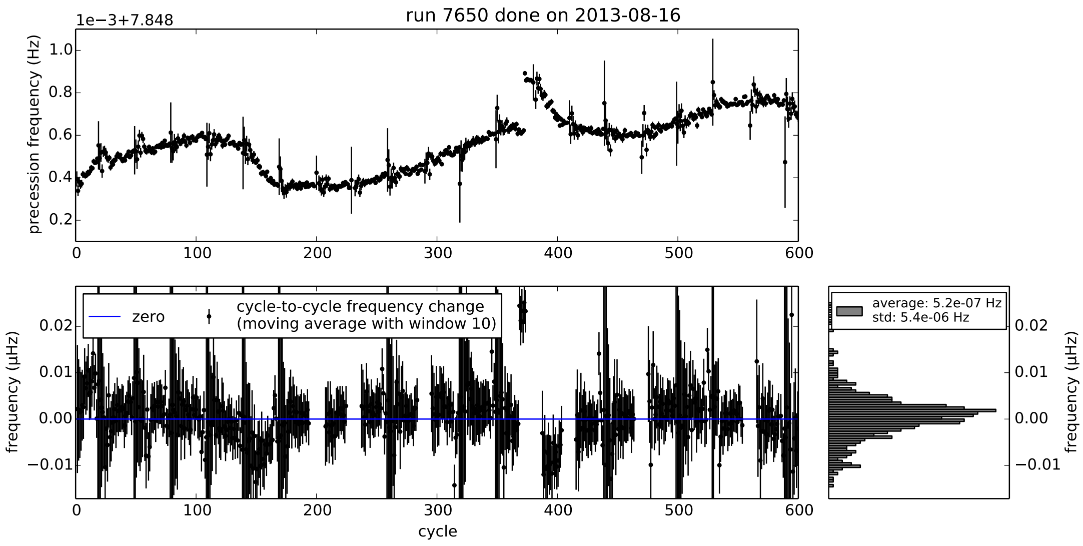


FIGURE 3.34: *Top panel: initial estimator of an average frequency in each cycle of run 7650, obtained with the* point-by-point *method with* two windows *(2 s and 15 s). Regular error–bar spikes occur for cycles performed immediately after changes of the high–voltage electrodes polarity. Lower left panel: cycle-to-cycle average precession frequency change, smoothed with a moving average with window size of 10 cycles. Lower right panel: distribution of the smoothed cycle–to–cycle frequency changes.*

Having determined the parameters describing the run, simulation can be performed in order to optimise sizes of the windows in the *two windows* method. The methodology was covered in Sec. 3.9.4. The point-by-point version of the *two windows* method was used (Sec. 3.10.3). In the simulations frequency of the generated signal is set to be close to the central frequency of the filter. It is done so to take into account only systematic effect related to frequency drifts. Basing on the plot (Fig. 3.35), it has to be decided which window size is to be considered optimal. In this case, size of the second window was chosen to be **130 s**. The first window is, as a rule of thumb, 20 times smaller. This choice gives drift–related systematic effect of $\approx \mathbf{2 \times 10^{-6}\,Hz}$ with precision of $\approx 4 \times 10^{-6}$ Hz.

The next step is to see how precision and accuracy of the analysis look, when mismatch between the frequency of the signal and the central frequency of the filter is included in the simulations. The results are shown in Fig. 3.36. The precession frequency in this run was very close to the central frequency of the filter (difference has the order of 10^{-3} Hz), so the accuracy difference for those two frequencies is negligible. Therefore, there is no additional systematic effect related to this mismatch.

To sum up, in analysis of the run 7650 precision of $\approx \mathbf{4 \times 10^{-6}\,Hz}$ is expected, with frequency drift–related systematic error of $\approx \mathbf{2 \times 10^{-6}\,Hz}$ and no systematic error due to mismatch of frequencies of the signal and the filter.

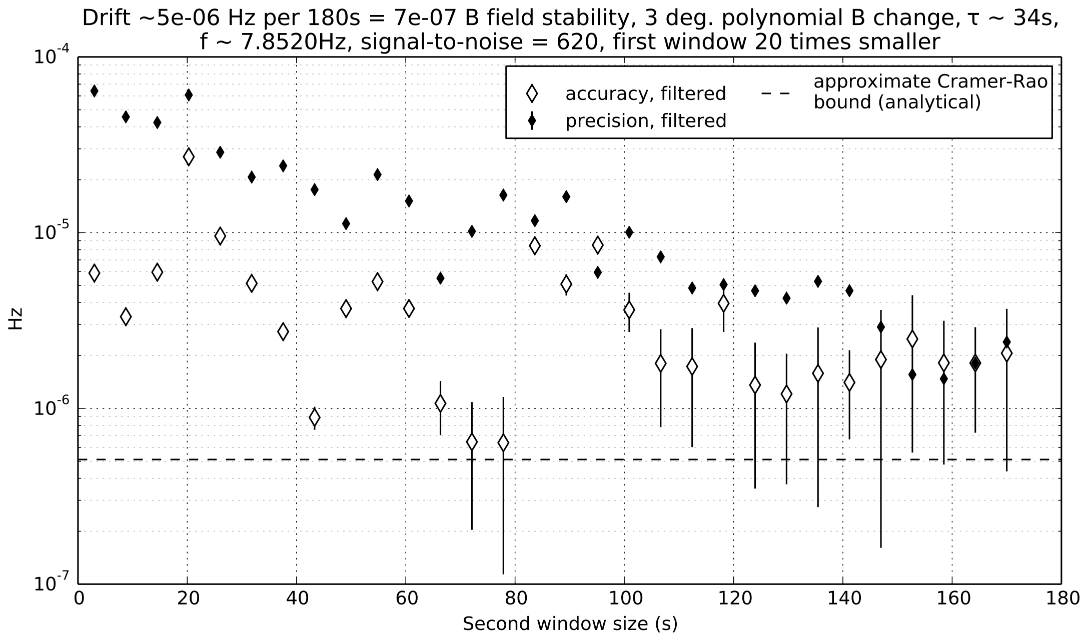


FIGURE 3.35: *Accuracy and precision of average precession frequency estimation, as a function of size of the second window (fist one being 20 times smaller). Simulations performed with the filter. The point-by-point version of the two windows method was used (Sec. 3.10.3).*

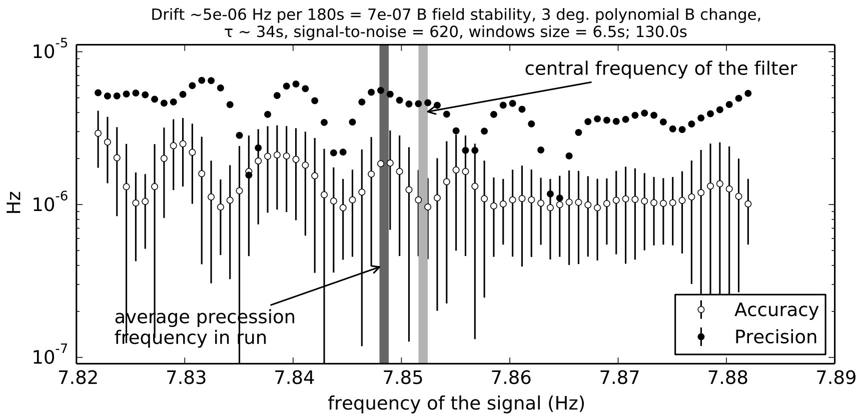


FIGURE 3.36: *Accuracy and precision of the average frequency estimation, as a function of a mismatch between the precession frequency and the central frequency of the filter.*

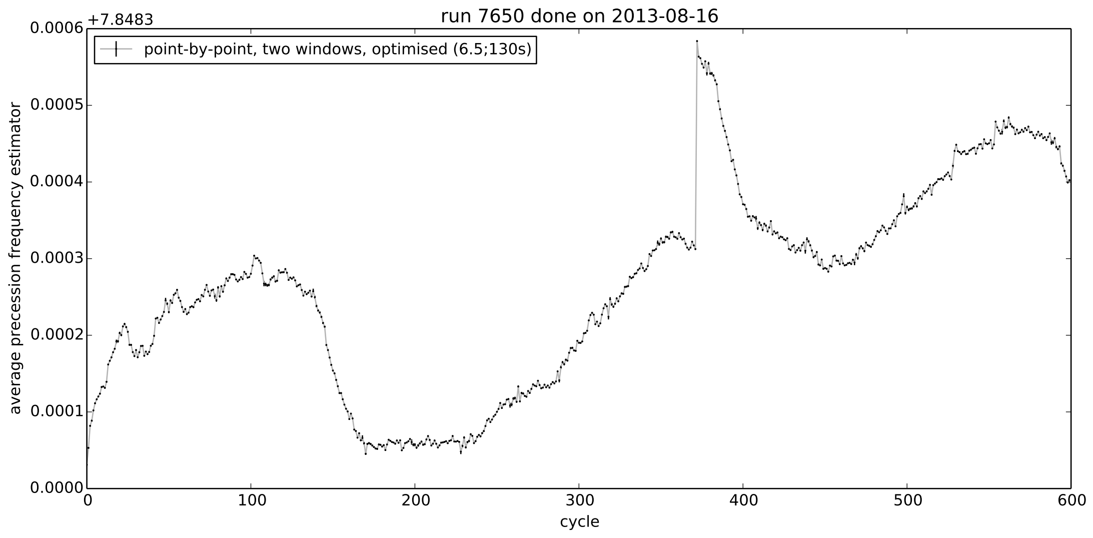


FIGURE 3.37: *Average precession frequency in each cycle of run 7650, analysed with the point-by-point method with two windows, with optimised sizes of windows.*

Finally, each cycle of the run is analysed with the method with optimised sizes of windows. Results are shown in Fig. 3.37. To illustrate the improvement, first 50 cycles of the run were analysed additionally with traditional two windows method with 15 s long windows, one with phase fitted at the end of the second windows and one with optimised sizes of windows. All are presented, together with the results of the on–line analysis for comparison, in Fig. 3.38. Note that the results given by the optimised traditional two *windows method* and its *point-by-point* version are essentially the same. However, there were cycles (usually after electrodes polarisation change), where the traditional one would fail to converge due to the second window being very long – 110 s. The results of the on–line analysis for these cycles form unnatural spikes. Meanwhile, the optimised *two windows* method in the *point-by-point* scheme always gave reasonable results.

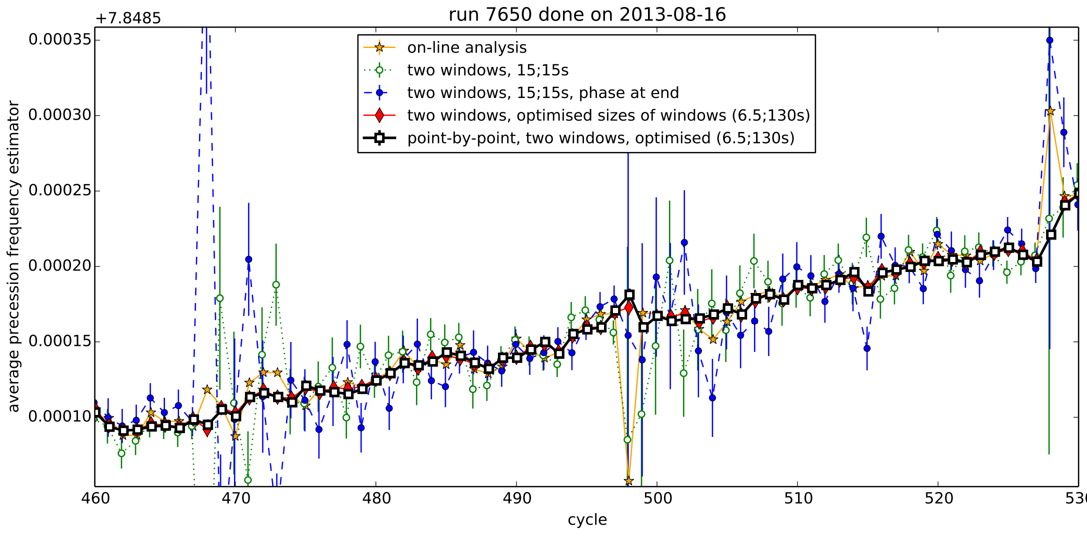


FIGURE 3.38: *Average precession frequency in first 50 cycles of the run 7650, as estimated with four off–line methods. The results of the on–line analysis are also shown for comparison.*

3.13.3 Analysis of run with long τ – run 4658

At times when run 4658 was carried out, the ^{199}Hg was considered to be performing exceptionally well, in contrast to those when the previously presented run 7650 took place. The process of the data analysis is illustrated with the same kind of plots as before:

1. Signal-to-noise distribution – Fig. 3.39. The distribution is far from being Gaussian. Still, the average of **144** is considered as representative.

2. Relaxation times distributions – Fig. 3.40. As representative for the run values $\tau_1 = \mathbf{12}\,\mathbf{s}$ and $\tau_2 = \mathbf{143}\,\mathbf{s}$ are taken.

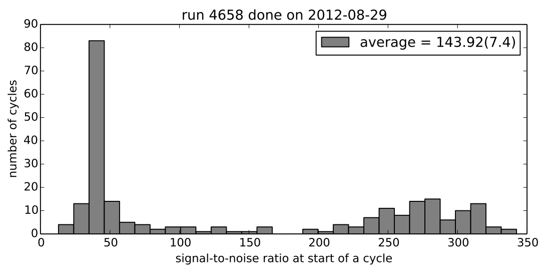


FIGURE 3.39: *Histogram of the comagnetometer signal–to–noise ratio for all cycles of run 4658.*

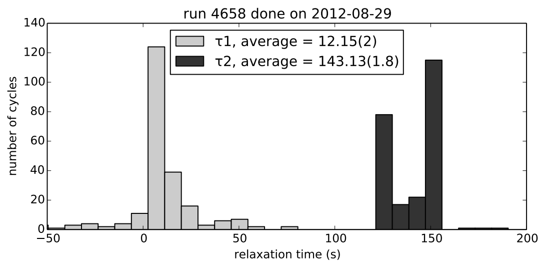


FIGURE 3.40: *Histogram of relaxation times of the comagnetometer signal for all cycles of run 4658.*

3. Average precession frequency in the run – Fig. 3.41. The value is **7.7645 Hz**.

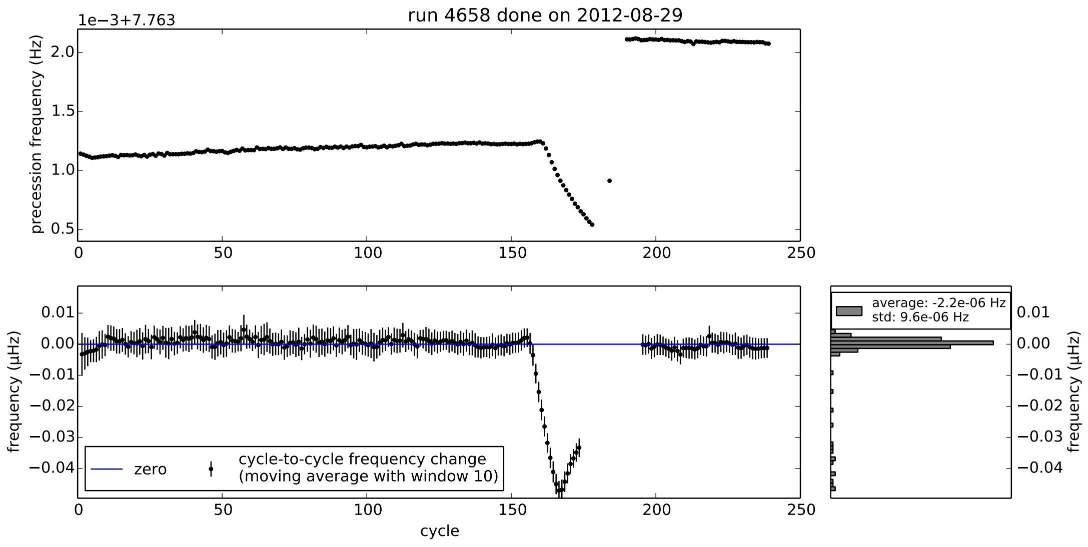


FIGURE 3.41: *As Fig. 3.34, but for the run 4658. A smaller influence of polarity changes is visible.*

4. Precession frequency drifts – Fig. 3.41. An estimated drift that may have occurred inside each cycle of the run is $\mathbf{9.6 \times 10^{-6}\,Hz}$.
5. Optimisation of windows sizes for the two windows method – Fig. 3.42. **60 s** was chosen as size of the second window, $\mathbf{60/20 = 3\,s}$ – as the first.
6. Accuracy with frequency mismatch – Fig. 3.43. From the plot it can be read, that the precision of $\approx \mathbf{1 \times 10^{-6}\,Hz}$ is expected. The additional accuracy loss (i.e. systematic effect) for frequencies occurring in the run is $\approx \mathbf{5 \times 10^{-7}\,Hz}$, in comparison to one expected with them being closer to the central frequency of the filter.
7. Calculation of the average precession frequency in each cycle of the run – Fig. 3.44.
8. Comparison of results given by 4 different methods of analysis for 50 cycles of the run – Fig. 3.45

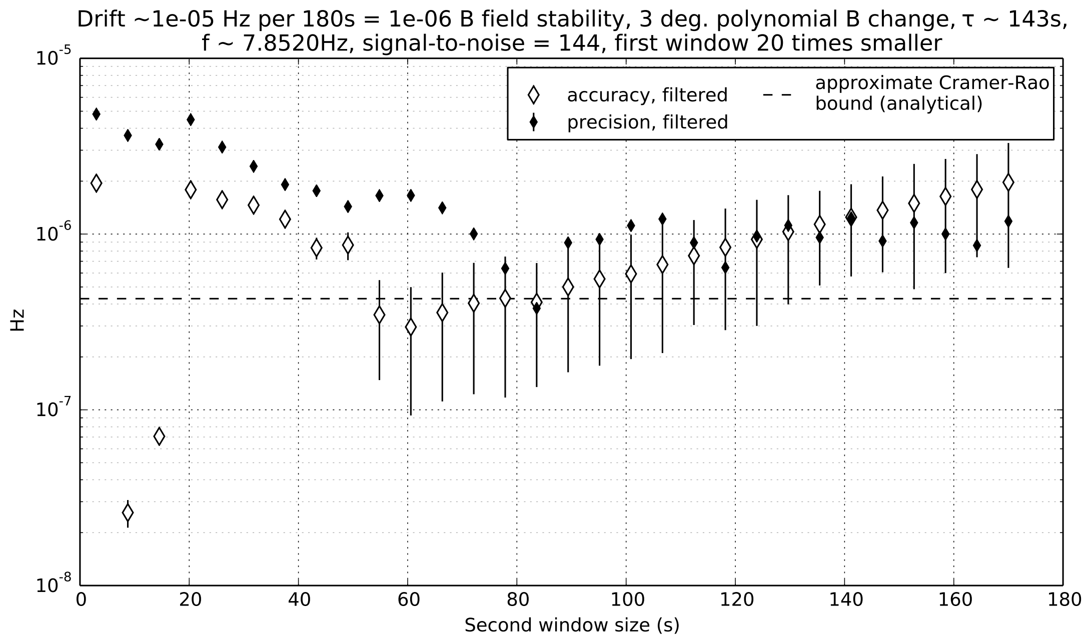


FIGURE 3.42: *As Fig. 3.35, but for the run 7658.*

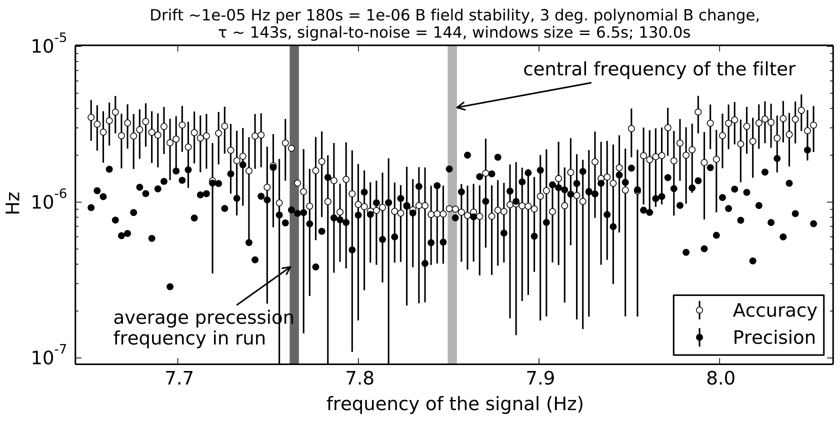


FIGURE 3.43: *Accuracy and precision of average precession frequency estimation, as a function of mismatch between precession frequency and central one of the filter.*

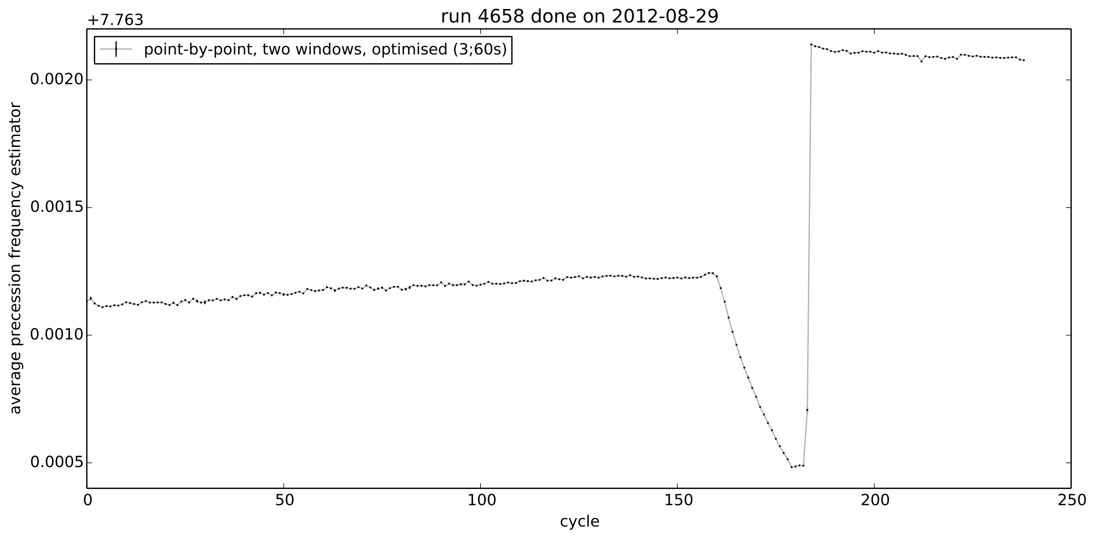


FIGURE 3.44: *Average precession frequency in each cycle of run 4658, analysed with point-by-point method with two windows, with optimised sizes of windows.*

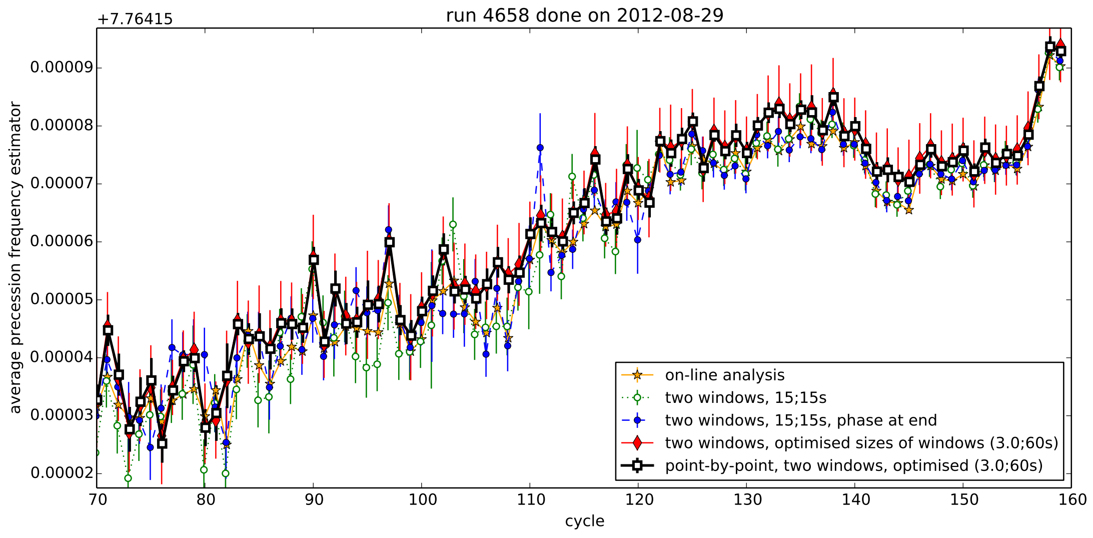


FIGURE 3.45: *Average precession frequency in first 50 cycles of the run 4658, as estimated with four methods. The results of the on–line analysis are also shown for comparison.*

Chapter 4

Design of a PXI–based DAQ and control system

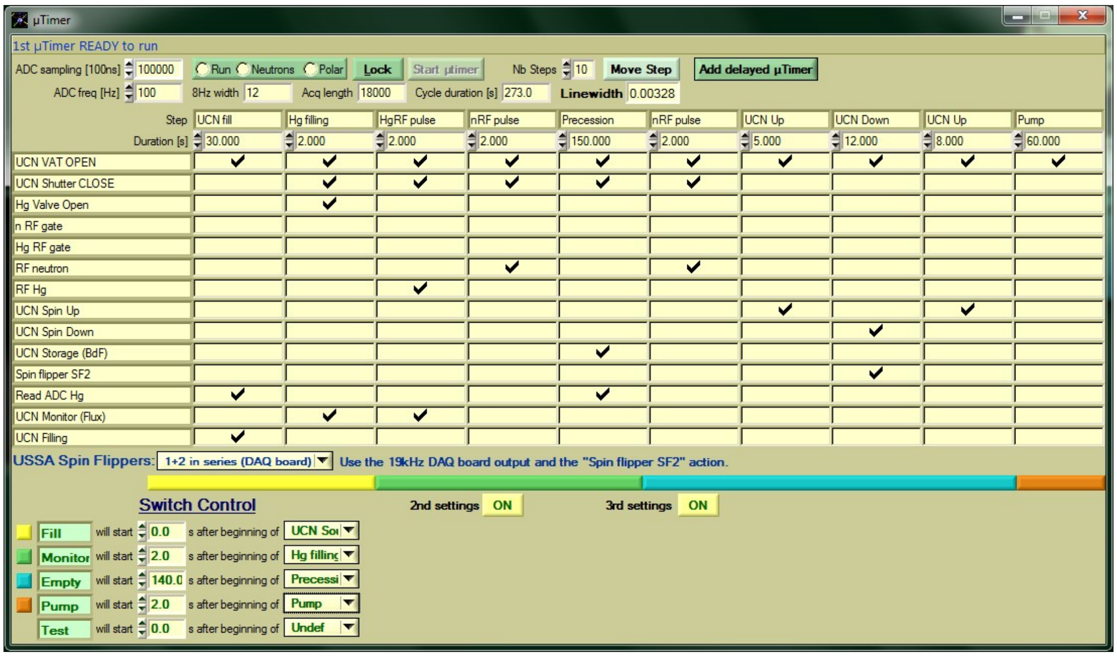
4.1 Introduction

A cycle is a fundamental concept in both the control of the experiment and the data acquisition design. A typical *cycle* is described in Section 2.4. Most important thing about *a cycle* is that it is precisely defined prior to its execution by specifying sequence of actions together with their timing (called a *schedule*). Each action has time specified when it should be executed, relative to *a trigger* — a marker which either comes from the ultracold neutrons source or is fired by software. *Cycles* are defined by shifters via user interface part of *AcqEDM* program (written by Zejma [9]). An example cycle definition is shown in Fig. 4.1.

After *a cycle* has been defined, its description is given to a device responsible for its execution. The device analyses it, prepares the experimental set-up, waits for the *trigger* and then runs the cycle. From the low level point of view the device is required to support four categories of operations:

TTL pulses, which require setting voltage an a wire in a binary manner — either *high* or *low*. They are used to control of the valves and for high–speed time markers. While valves can be controlled with 100 ms precision, the high–speed time tags are required to be precise on the level of hundreds of µs.

DAC output involving setting arbitrary voltage between two wires. Used for driving current in coils to generate the ELF magnetic field pulses. This has the most strict timing requirement — generation of consecutive samples has to be timed with stability of at least 10^{-9}.



µTimer

1st µTimer READY to run

ADC sampling [100ns] 100000 · Run · Neutrons · Polar · Lock · Start µtimer · Nb Steps 10 · Move Step · Add delayed µTimer

ADC freq [Hz] 100 · 8Hz width 12 · Acq length 18000 · Cycle duration [s] 273.0 · Linewidth 0.00328

Step	UCN fill	Hg filling	HgRF pulse	nRF pulse	Precession	nRF pulse	UCN Up	UCN Down	UCN Up	Pump
Duration [s]	30.000	2.000	2.000	2.000	150.000	2.000	5.000	12.000	8.000	60.000
UCN VAT OPEN	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
UCN Shutter CLOSE		✓	✓	✓	✓	✓				
Hg Valve Open		✓								
n RF gate										
Hg RF gate										
RF neutron				✓		✓				
RF Hg			✓							
UCN Spin Up							✓		✓	
UCN Spin Down								✓		
UCN Storage (BdF)					✓					
Spin flipper SF2								✓		
Read ADC Hg	✓				✓					
UCN Monitor (Flux)		✓	✓							
UCN Filling	✓									

USSA Spin Flippers: 1+2 in series (DAQ board) Use the 19kHz DAQ board output and the "Spin flipper SF2" action.

Switch Control · 2nd settings ON · 3rd settings ON

🟨	Fill	will start	0.0	s after beginning of	UCN So
🟩	Monitor	will start	2.0	s after beginning of	Hg filling
🟦	Empty	will start	140.0	s after beginning of	Precessi
🟧	Pump	will start	2.0	s after beginning of	Pump
	Test	will start	0.0	s after beginning of	Undef

FIGURE 4.1: *An example* cycle *as defined in the* AcqEDM *program. The leftmost column contains various actions that may be executed. Two uppermost rows define* steps — *consecutive periods of time after the trigger event.* Step *is defined by its name (top row) and duration (second from top row). Each* action *may be performed in each* step, *which is marked by a tick, or not.*

ADC input, which is a measurement of a voltage between two wires. It is used for readout of the photomultiplier voltage, which is part of the ^{199}Hg comagnetometer. Timing requirements are as strict as for the DAC output.

digital communication, mainly with the user interface computer. These operations need not to be timed more precisely than 100 ms.

It is important to realise timing constraints that system responsible for execution of *cycles* should have. These requirements create need for a **real–time system**, which can be defined, in simple words, as *a system in which when a result comes is at least as important as whether it is correct.* Real–time systems are also *time–deterministic*, which is very important as the measurement should be as repetitive as possible.

4.2 The daqBoard

The current system responsible for realisation of cycles, called the *daqBoard*, has been used since the beginning and is already in its third revision. It was developed and build by professional electronic engineers working at LPSC (Laboratoire de Physique Subatomique et de Cosmologie) in Grenoble, France and is described in publication by Bourrion et al. [13]. The device, shown in Fig. 4.2, is a custom etched printed circuit board (PCB) with the most important chip being an FPGA (*Field Programmable Gate*

Array). These technologies ensure real–time capabilities, reliability and relatively low production cost of the actual device. However, this it at the cost of a low–level design requiring very specialised technical knowledge. This fact, combined with the engineers not working full–time on the project, causes the development cycle of this solution to be rather slow. This is a major drawback, as the current *nEDM* experiment is intended largely as a research and development polygon for the next–generation one – the *n2EDM* (described by Zejma [14]) and, as such, is often upgraded on a monthly basis. Therefore, the current control system, *daqBoard*, is often a limiting factor in the development of the set-up.

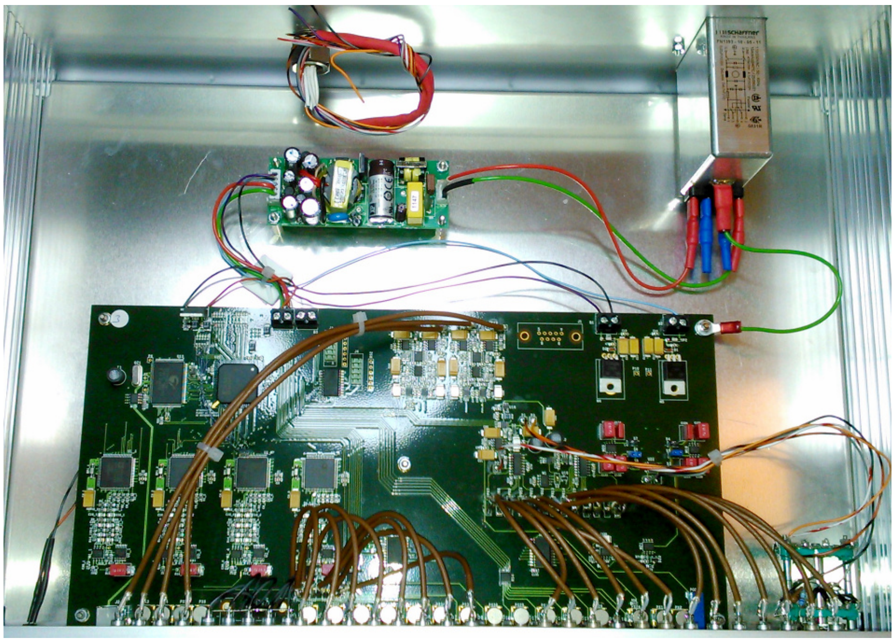


FIGURE 4.2: *A photograph of the third revision of the daqBoard.*

4.3 PXI system

Limitations of the current system motivated proposal of an another system with two main features in mind: **modularity** and **shortness of development cycle**. Both are met by the *PXI* system, which is very well described by National Instruments (the original designer of PXI specification) [15]:

PCI eXtensions for Instrumentation (PXI) is a rugged PC-based platform that offers a high-performance, low-cost deployment solution for measurement and automation systems. PXI combines the Peripheral Component Interconnect (PCI) electrical bus with the rugged, modular Eurocard mechanical packaging of CompactPCI and adds specialized synchronization buses and key software features. PXI also adds mechanical, electrical, and software features that define complete systems for test and measurement, data acquisition, and manufacturing applications.

In 2005, additionally the *PXI Express* standard was released, in which the PCI bus was changed to PCI Express, improving bandwith by a factor of 45. The new standard added also another improvements, again described by National Instruments [15]:

Building on PXI capabilities, PXI Express provides the additional timing and synchronization features of a 100 MHz differential system clock, differential signaling, and differential star triggers.

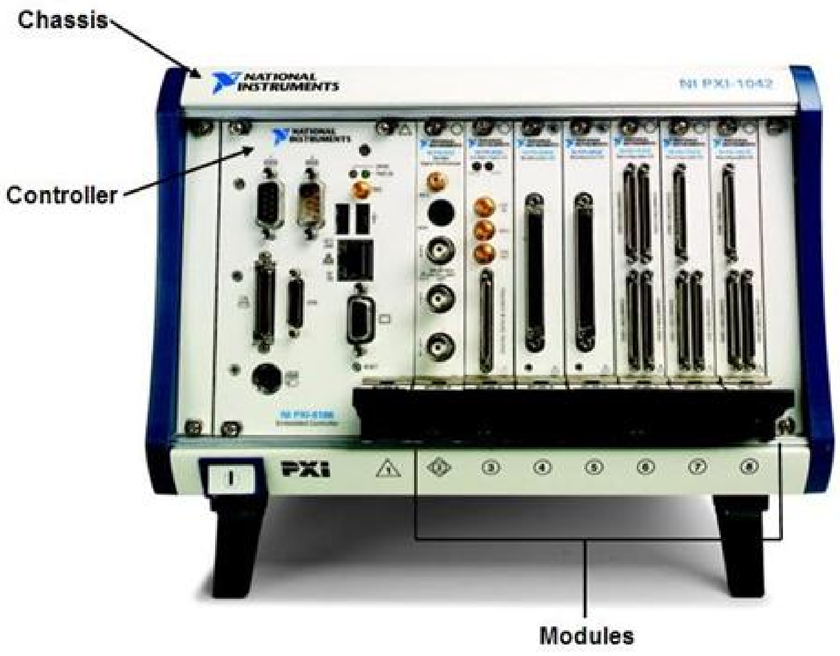


FIGURE 4.3: *Front of 8-Slot PXI Chassis manufactured by National Instruments, model PXI-1042. All slots are filled with modules. Source: [15]*

The PXI system can be divided into three parts:

PXI chassis, pictured in Fig. 4.3, serves as a rack for *modules.* Chassis provides PCI bus (*Peripheral Component Interconnect*) and/or, in case of PXI Express — PCI Express bus, to which all modules connect. This allows them to exchange data with bandwidth of 123 MB/s. The PCI bus is accompanied by additional trigger and clock buses, which allow a very accurate synchronisation between modules. A scheme of these lines is presented in Fig. 4.4. All the buses have a common name: *PXI backplane.* A *chassis* contains also a power supply and a reference clock, which can be distributed among modules.

Modules (or rather their application) are the actual purpose of the PXI system. There is a tremendous variety of available modules, including: serial ports, General Purpose Input–Output (GPIO), oscilloscopes, Analogue to Digital Converters (ADCs), Digital to Analogue Converters (DACs), FPGAs, relays, current sources and many more. They all plug into the *backplane* and thus can exchange data and synchronise each other.

PXI controller is a very special kind of module, which is a PC–compatible computer build with industrial standard components. On one end it plugs into a *PXI backplane* and on the other it has standard interfaces line Ethernet, USB ports and display port. It runs either Microsoft Windows or National Instruments LabVIEW Real–Time operating system and is usually programmed in LabVIEW or LabWindows/CVI. One has a full control over all modules in chassis from within the software running on the *controller*, which allows one to perform very sophisticated data acquisition and control, all in a familiar PC development environment. The use of the LabVIEW Real–Time operating system additionally makes software, and thus performed tasks, time–deterministic.

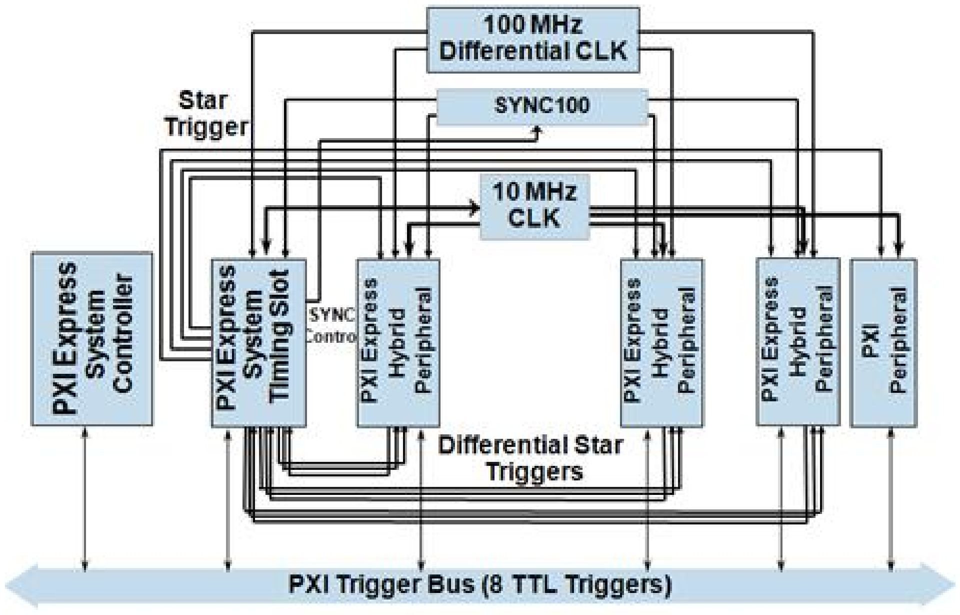

100 MHz Differential CLK
Star Trigger
SYNC100
10 MHz CLK
PXI Express System Controller
PXI Express System Timing Slot
SYNC Control
PXI Express Hybrid Peripheral
PXI Express Hybrid Peripheral
PXI Express Hybrid Peripheral
PXI Peripheral
Differential Star Triggers
PXI Trigger Bus (8 TTL Triggers)

FIGURE 4.4: *PXI Express timing and triggering buses. Source: [15]*

Typically, all external communication with the controller happens over TCP/IP protocol. Aside from all the custom transmissions that may happen in this way, the controller always runs an under–the–hood service designated for an upload of programs, execution control and debugging. Thanks to it, PXI is very efficiently programmed not only by working directly on the embedded controller, but also with a use of any computer, provided that it is connected the same network as the crate. Actually, when developing a project for PXI on a laptop, compiling and running programs locally is indistinguishable from running them remotely on a PXI controller. This includes all debugging features, like break–points, pausing an execution, stepping thorough program or placing *probes* used to peek on variables. Finished program is *deployed*, that means loaded into the controller together with all its dependencies. After a successful *deployment*, when crate is powered on it automatically boots and executes the deployed application.

4.4 Software design

4.4.1 Technologies used

In creation of a PXI–based control system main difficulty falls in design and implementation of the software to run on the controller. It was decided, that it will be written in LabVIEW G programming language (example source code is presented in Fig. 4.5) and run on LabVIEW Real–Time operating system. Because numerous tasks need to be performed, often simultaneously, software employs a number of threads, running in parallel rather than being linear. Such construction is encouraged by LabVIEW and also allows for better CPU utilisation, as processors in PXI controller have as many as 4 or 8 cores and can thus run even 4 or 8 thread simultaneously.

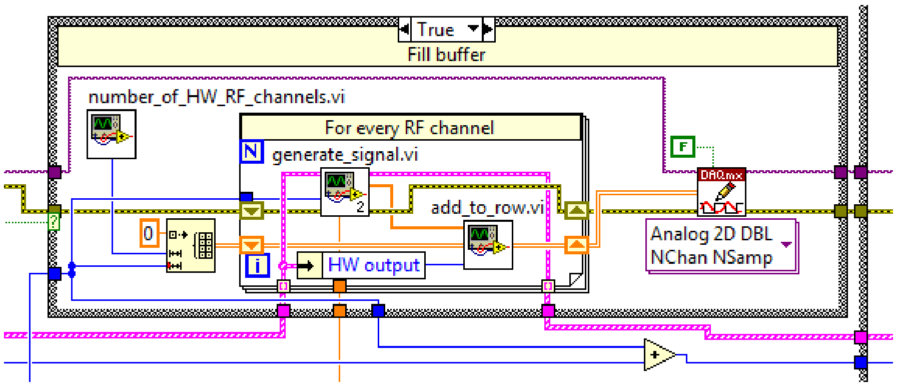


FIGURE 4.5: *A fragment of LabVIEW G language source–code. Rather than being understood, it is meant to give a glimpse of how programming in this nontraditional programming language looks like. Functions and constants are represented as blocks, while variables propagate through wires. Case structures and for loops are coded with frames.*

With such complicated structure, timing and synchronisation requirements have to be met, which is possible with use of mechanisms provided by the real–time operating system. Of the many kinds of such mechanisms, three ore used in the presented software. These are:

Timed structures (loops and sequences), which are fundamental to all real–time operating systems and allow an advanced timing control of execution. Each timed structure is an individual thread in a system. The code inside such structure has a defined time–window in which it should execute, by having a *deadline* specified (loops additionally a period). A special process, *LabVIEW scheduler*, is then responsible for switching threads such, that all deadlines are met. If for some reason they are not, a flag is raised and appropriate fall–back procedure may also be implemented. Timed structures have also advanced options for execution control, most notably *priority*, defining which structures are delayed first when not all deadlines can be met.

Rendezvous, which is one of the basic mechanisms for thread synchronisation. It is a set of blockades, located in different threads, which block thread execution until all blockades in a given set are reached. Therefore, instructions synchronised by a *rendezvous* start simultaneously.

Message queues, which allow for an asynchronous data exchange between threads. A piece of data may be put on a *message queue* by a thread, which can then immediately continue its execution. The piece is later taken off the queue by another thread.

4.4.2 Structure of the software

There are 7 threads in the implemented software:

1. **Control thread**, which manages all others by communicating with them over message queues.
2. **General scheduler**, which is a basic mechanism of performing actions according to a schedule of a cycle
3. **TCP/IP communication thread**

The remaining three are control threads dedicated to particular hardware actions:

4. **DAC control thread**, responsible for generation of oscillating pulses of current in coils (which create then the ELF pulses)
5. **ADC control thread**, which acquires and processes the signal from ^{199}Hg co-magnetometer
6. **TTL thread**, which sends TTL pulses
7. **Hardware trigger thread**, which executes only when triggered by a hardware trigger line

When the system prepares for a cycle execution its schedule is analysed by the control thread and then passed on to the general scheduler, which can execute it with $\approx 1\,\mathrm{ms}$ accuracy. For example, if schedule says that $10\,\mathrm{s}$ after the trigger a TTL pulse should be sent, then at this time the general scheduler sends a message (via a message queue) to the TTL thread, telling it to send the pulse, which it does. This mechanism provides only ca. $\approx 1\,\mathrm{ms}$ accuracy, which is fine for some actions, but not for all. Some things, like ADC readout, cannot be realised in this scheme and these are passed by the control thread directly to control loops of devices. This will be described in detail in the next three sections.

4.4.3 Assurance of phase coherence of ELF pulses

Lying at the basis of the nEDM experiment *Ramsey method of oscillatory fields* involves applying two oscillating pulses of magnetic filed, with phases coherent with one another. If the *general scheduler* was used for generation of the pulses, the coherence would not be satisfactory. Nevertheless, it can be greatly improved if another technique is employed.

Signal generation with a DAC is handled in PXI by a structure called a *task*. It is an abstract object, which represents the device and various parameters of its operation. It is used to configure the device, mediates in data exchange with it and handles starting and stopping the actual generation. When defining a *task*, a total number of samples to be generated, N, can be specified, and a cyclic buffer (typically much smaller than N) is created in the RAM of a PXI controller. When the *task* is run the device takes the samples from the buffer and generates them, each sample being triggered by a reference frequency source clock. Still, software running on the controller has to load new samples to the cyclic buffer at appropriate pace, because if the DAC runs out of samples before it has generated all N points *task* is immediately terminated and an error is thrown. It follows that within one *task* timing, phase coherence of the generated ELF pulses is as precise as reference frequency source. If an atomic clock is used as the source, timing precision and stability have the order of 10^{-11}. But still, a *task* is started by software, which has at best 10^{-6} timing capabilities. Therefore, the best achievable coherence if two different tasks are used for the two pulses is only of the order 10^{-6}.

For the above reasons only one ADC *task* is created per cycle, lasting from the start of the first pulse to the end of the last one. This means that for a long time, during free precession of the neutrons, when no pulses are desired, zeroes have to be continuously fed into the cyclic buffer and generated by the ADC. However, this allows phase coherence of all pulses to be on the same level as precision of the atomic clock.

4.4.4 ADC readout handling

In the daqBoard, the comagnetometer signal is routed through an analogue filter. It is done so not only to reduce noise, but also to separate AC and DC components before they are digitised. This allows one to use the full 16-bit range of the ADC in daqBoard to measure the more interesting AC component, while DC is measured with a separate digitiser. As influence of the filter on data is be controversial (see Sec. 3.9.5), it is often considered to alter its parameters. Unfortunately, the analogue filter has fixed parameters defined by electrical components soldered into the electronics.

In the alternative PXI design signal does not have to be filtered before digitisation thanks to much more precise, 24-bit ADC. Filter is still required, but in the PXI it can be applied on already digitised data. *Digital filter*, as it's called, is considered superior to its analogue equivalent for two reasons. Firstly, as it is fully implemented in software, its parameters can be freely changed. Even raw, non-filtered data can be recorded and filtering applied only during an off–line analysis. Secondly, analogue filters are known to introduce a phase distortion to the filtered signal, which can be avoided in digital filters. Filters that do not exhibit such behaviour are said to have a *linear phase*.

The aim of the measurement of the comagnetometer signal is to find its frequency, which is equal to ca. 8 Hz. Therefore, even though the signal is digitised at maximal available rate, it is down–sampled to 100 Hz. It was shown by Chibane et al. [6] that denser sampling does not improve precision of frequency estimation. Down–sampling takes place before filtering in order to save computing power.

Usually, the signal of the comagnetometer is not interesting throughout the whole cycle, but only during two periods: first, before mercury atoms are injected to the precession chamber — to measure noise of the apparatus; second, during free precession of the neutrons — to collect information about the magnetic field. In order to make it easy to extract only the desired parts of the readout, a concept of *ADC buffers* is introduced. Data is read, down–sampled and filtered continuously, but it is up to an author of a cycle schedule to define, when it will be stored and into which of 8 available buffers. Samples stored in the buffers can be then retrieved at any time, even after the cycle is ended.

4.4.5 Triggering

Trigger in this system is a time marker signalising that cycle should start. In general it is possible to use a hardware signal, however, it was decided that software–based triggering will be used. Such approach greatly simplifies structure of the system while retaining accuracy relative to the trigger of the order of tens of microseconds..

When the system prepares for an execution of a cycle, all threads are instructed to perform actions according to schedule, but do not start yet, as they are blocked by a *rendezvous* functions. The *rendezvous* is completed when *trigger timed structure* is executed, which simultaneously unblocks all waiting threads. The *trigger timed structure* is a special loop which is not timed by a clock, but a hardware input – the trigger line. This minimises delay between the hardware trigger and release of the *rendezvous*, without need of resource–consuming mechanisms such as polling or an active blockade.

4.5 Interface of the system

4.5.1 Goals

Interface of the software was designed to achieve two goals: **compliance with scheme of the data acquisition system of the next generation n2EDM experiment** and

simplicity. The coarse shape of the data acquisition system of the n2EDM experiment was established on the DAQ meeting in October 2013 in Cracow. It was decided, that ideally all DAQ subsystems would be controlled by text commands, loosely based on the SCPI standard, sent over TCP/IP. Other data could be transmitted using other, possibly dedicated protocols, but still over an Ethernet cable. Design of the presented here interface tries to follow these decisions not only to allow it to later seamlessly transfer into the n2EDM DAQ, but more importantly to serve as testing platform for a such solution.

Simplicity was desired at level such, that it would take at most several hours to write from scratch a script capable of communicating and controlling the PXI system. It is an answer to frequent need of performing one–off measurements, designated for tuning the apparatus or troubleshooting weird behaviours. Such measurements could be easily automated using the proposed script command language.

4.5.2 Commands

The PXI controller is running a TCP/IP server (port 6666), listening for text commands. The commands are organised into a tree structure, where the root is the name of the system: `UTIMER` (for micro–timer). The second level of the command tree represents subsystems: `TTL` (TTL pulses), `ELF` (generation of ELF pulses), `ADC` (readout of the ^{199}Hg comagnetometer), `TRIGGER` and `SCHEDULE`. The tree is traversed by giving names of the nodes, root first, separated by colons. For example, the following message commands the system to send an ELF pulse on the channel 0:

```
UTIMER:ELF:ELF0
```

Some commands accept parameters. They are given after space. For example, the TTL line 0 may be either set high, low or commanded to send a short pulse (i.e. change its state twice):

```
UTIMER:TTL:TTL0 1
UTIMER:TTL:TTL0 0
UTIMER:TTL:TTL0 PULSE
```

Messages with a question mark at the end are queries. The most basic one is query on the system's identification string:

```
UTIMER:ID?
```

After receiving such message the system replies on the same TCP/IP connection:

```
PSI_nEDM_PXI_MR
```

In fact all messages are replied: queries with an answer, commands are echoed (without parameters).

Appendix B contains a full tree of implemented commands, each with description.

4.5.3 Examples

The interface was designed to be easily used in simple scripts. Two examples in Python3 language are presented, serving as demonstration, as well as covering typical usage scenarios.

First script, listing 4.1, uses only the `socket` module, which is part of the standard Python3 library, distributed always with the interpreter. The script covers following topics:

1. Two simple functions are defined to simplify communication with the PXI system.
2. A simple query is made, `uTimer:ID?`, and answer is checked to be equal `PSI_nEDM_PXI_MR`
3. Parameters of an ELF pulse are set.
4. A simple schedule is defined, consisting of three actions: a TTL pulse immediately after the trigger, a 2 s long ELF pulse starting 5 s after the trigger and the same pulse again 185 s after trigger.
5. The trigger is requested, leaving the system in a state such, that it executes set schedule immediately upon receiving a trigger.

The second script, listing 4.2, makes use of two utility functions defined in the first script. The program demonstrates powerful scripting capabilities of the designed interface for the PXI system. This short, 32–line script, performs an automatic scan of a two–dimensional parameter space: the duration and the amplitude of an ELF pulse. Note that automation, here a nested `for` loop, is provided solely by mechanisms of the programming language, not by the interface of the PXI system.

```python
import socket


def open_connection(ip, port=6666):
    """Open connection to ip on port and return socket object."""
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    server_address = (ip, port)
    s.connect(server_address)

    return s


def send_receive(s, message, delimiter='\r\n'):
    """Send message over socket s and listen to reply until delimiter."""
    s.sendall(message)

    n = len(delimiter)
    rmessage = ''
    while rmessage[-n:] != delimiter:
        rmessage += str(s.recv(1))

    return rmessage[:-n]


if __name__ == '__main__':
    s = open_connection('192.168.1.2')

    # check if communication works
    if send_receive(s, 'uTimer:ID?') != 'PSI_nEDM_PXI_MR':
        print('Communication NOT OK!')

    # set parameters of an ELF pulse
    send_receive(s, 'uTimer:ELF:ELF0:frequency 8')
    send_receive(s, 'uTimer:ELF:ELF0:amplitude 1.2')
    send_receive(s, 'uTimer:ELF:ELF0:envelope sine')
    send_receive(s, 'uTimer:ELF:ELF0:hardwareOutput 0')

    # Define a simple schedule
    send_receive(s, 'uTimer:schedule:set "'
                 '0 uTimer:TTL:TT0 pulse;'
                 '5 uTimer:ELF:ELF0 2;'
                 '185 uTimer:ELF:ELF0 2"')

    # Request trigger - schedule will execute when it comes
    send_receive(s, 'uTimer:trigger:request')
```

LISTING 4.1: Python3 script which: 1. defines two simple functions to communicate with the PXI DAQ system. 2. Performs a simple query and checks the answer. 3. Sets parameters of an ELF pulse. 4. Defines a simple schedule. 5. Requests a trigger.

```python
from time import sleep
from PXI_communication import open_connection, send_receive

s = open_connection('192.168.1.2')

send_receive(s, 'uTimer:ELF:ELF0:frequency 8')
send_receive(s, 'uTimer:ELF:ELF0:envelope sine')
send_receive(s, 'uTimer:ELF:ELF0:hardwareOutput 0')

# scan two dimensional space of pulses duration and amplitude
for duration in [0.5, 1, 1.5, 2, 2.5, 3]:
    for amplitude in [0.1, 0.5, 1, 1.5, 2, 3]:
        send_receive(s, 'uTimer:ELF:ELF0:amplitude %.1f' % amplitude)

        send_receive(s, 'uTimer:schedule:set "'
                     '0 uTimer:TTL:TT0 pulse;'
                     '5 uTimer:ELF:ELF0 %d;'
                     '185 uTimer:ELF:ELF0 %d;'
                     '200 uTimer:schedule:end"' % (duration, duration))

        send_receive(s, 'uTimer:trigger:request')

        # wait for trigger to come
        while send_receive(s, 'uTimer:trigger:came?') != '1':
            sleep(1)

        # ignore next trigger - run only once
        send_receive(s, 'uTimer:trigger:mask 0')

        # wait for the cycle to finish
        while send_receive(s, 'uTimer:schedule:end?') != '1':
            sleep(1)
```

LISTING 4.2: Python3 script which performs an automatic scan of two–dimensional parameter space: ELF pulse duration and amplitude. Functions defined in listing 4.1 are used.

4.6 Tests

4.6.1 Introduction

Because the PXI system will play a crucial role in the experiment, it is necessary to test its technical specifications. In particular, two parameters characterising generation of the ELF pulses are of most importance: **noise** and **frequency stability**.

The latter is achievable on satisfactory level only with a synchronisation to an atomic clock. The architecture of the used PXI chassis supports this very well via a common 10 MHz `PXI_CLK10` bus with external synchronisation input. The bus is distributed among all modules in a chassis. Devices then use the *phase locked loop* technique to synchronise their internal clocks with this signal. The bus is normally driven by a quartz clock with, according to manual, 25 ppm stability. However, if external source is plugged, it is used instead, unless crate detects that it differs by more than 100 ppm from the internal quartz clock. Unfortunately no indicator of rejection of external source was found, posing a question whether the crate accepts the reference source at all.

Additional tests were needed to confirm two statements:

1. Devices can indeed synchronise their clocks to the `PXI_CLK10` bus and necessary know-how is mastered.
2. The `PXI_CLK10` accepts external reference frequency source, e.g. an atomic clock.

Even if the above statements are confirmed, synchronisation to an external atomic clock does not guarantee frequency stability of the generated signal on the same level. There still may be technical faults that may cause, for example, DAC to miss a sample. Therefore, stability had to be additionally measured by itself.

Also, as PXI is a modular system, there are several modules that could be used as ADC and DAC. The best suitable configuration had to be chosen, relying not only on manufacturer's specification, but also on dedicated measurements assessing performance in usages that these devices are to put to.

Another interesting parameter is timing accuracy of TTL pulses generated by the system.

4.6.2 Methodology

To assess the stability, an oscillating signal was generated on a DAC and it was measured on an ADC, both separate devices sitting in one PXI crate. Period of the signal was chosen to be an integer multiple N of ADC sampling. Therefore plotting only every Nth point should yield samples having exactly the same voltage — a horizontal line (see

Fig. 4.6). Small frequency drifts will tilt the line, larger will turn it into sine. Distribution of the points around the line reflects noise of the system. The method was proposed by Bison [16].

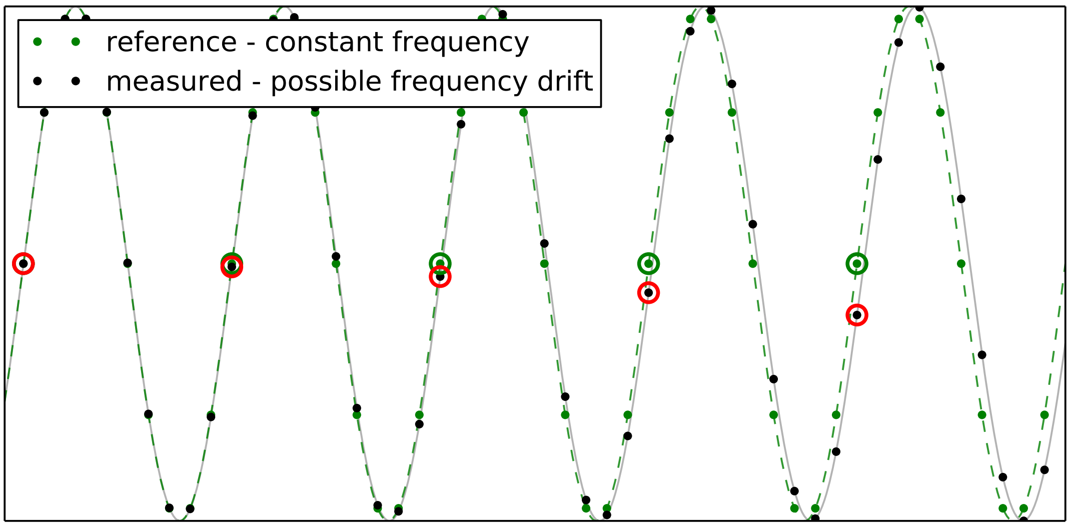


FIGURE 4.6: *Method used to test frequency stability. An oscillating signal is generated such that its period is an integer number N of samples. In the ideal case every Nth point will be exactly at the same level. Red points represent the case with frequency drift — they lie on a titled line, which can turn into a sine in case of very large drifts.*

Additionally, an effect which mimics frequency drift had to be taken into account: DC level drift. It still may occur, even though both generation and measurement were carried out differentially. It was be easily removed by applying to the measured signal a high–pass filter with around 0.5 Hz cut-off frequency (the signal being of the order of 10 Hz).

4.6.3 Synchronisation test

All results are juxtaposed in Table 4.1, and two of them are presented in detail in Figs. 4.7 and 4.8. The fact that the drift gets five orders of magnitude smaller when the devices synchronise to the `PXI_CLK10` bus, in comparison to no synchronisation whatsoever, proves the first of the statements from Sec. 4.6.1: the devices can synchronise their clocks to the `PXI_CLK10` bus. The second statement (the PXI CLK10 accepts external reference frequency source) was proved by another two measurements, both with DAC synchronising its clock to the `PXI_CLK10` bus and ADC having atomic clock plugged as reference frequency source via a front-face terminal. During the first measurement an atomic clock was plugged also to the `PXI_CLK10` bus, during second one — was not. Five orders of magnitude worse result in the second measurement leaves no doubts about truth of the second statement.

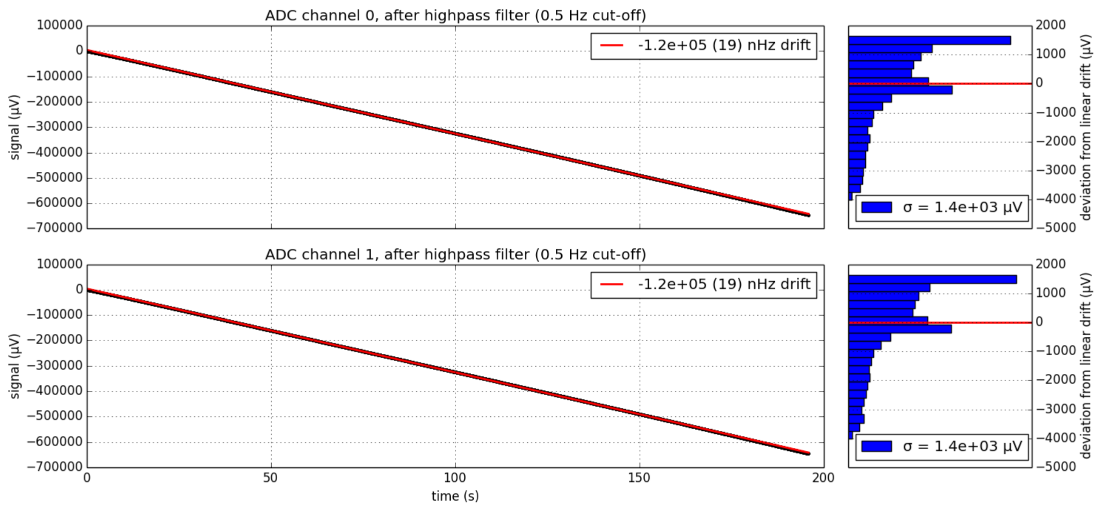


FIGURE 4.7: *Results of the frequency stability measurement of the PXI system, with ADC and DAC using their internal frequency sources. Plots on the left hand side show every Nth measured sample and linear fit to them. With no frequency drift it would be a horizontal line, but a clear skew is visible corresponding to 0.1 μHz drift. Histograms on the right show distribution of the points around the fitted line. Signal was generated and recorded on two ADC and two DAC channels simultaneously.*

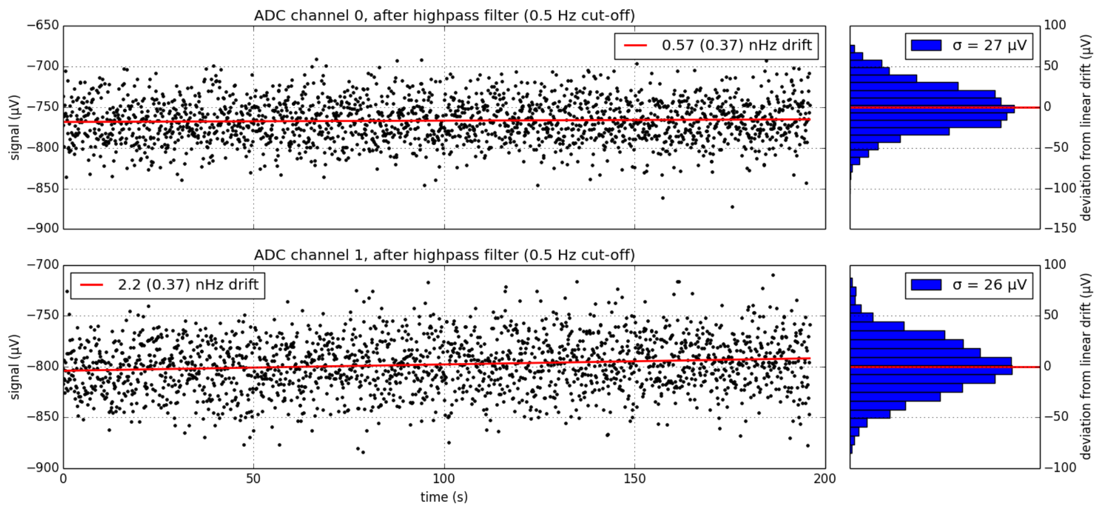


FIGURE 4.8: *As above, but with ADC and DAC synchronising their clocks to atomic clock via common* `PXI_CLK10` *bus. Fact that measured points form a horizontal line shows that no frequency drift was detected.*

DAC reference clock	ADC reference clock	measured frequency drift	
none	none	$117.00 \pm 0.02\,\mu\text{Hz}$	(Fig. 4.7)
`PXI_CLK10` bus	`PXI_CLK10` bus	$2.1 \pm 0.4\,\mu\text{Hz}$	
`PXI_CLK10` bus	atomic clock via external terminal	$158.000 \pm 0.009\,\mu\text{Hz}$	
atomic clock via `PXI_CLK10` bus	atomic clock via external terminal	$2.0 \pm 0.4\,\mu\text{Hz}$	(Fig. 4.8)

TABLE 4.1: *Juxtaposition of measured frequency drifts for various clocks configurations.*

4.6.4 Comparison of ADC modules

FIGURE 4.9: *National Instruments PXI-4461 module with two ADC and two DAC channels with independent timing.*

FIGURE 4.10: *National Instruments PXI-5922 module — two-channel ADC.*

A wide range of available ADC and DAC PXI modules, basing on manufacturer's catalogue, was narrowed down to two devices:

National Instruments PXI-4461 (Fig. 4.9) with:

- 2 channels of 24 bit Δ-Σ **ADC** with speed up to 204.8 kS/s (kilosamples per second)
- 2 channels of 24 bit Δ-Σ **DAC** with speed up to 204.8 kS/s, capable of supplying up to 16 mA of current.

National Instruments PXI-5922 (Fig. 4.10) with 2 channels of 24 bit Δ-Σ **ADC** with speed up to 500 kS/s

All channels in both modules have independent timing, according to choice of the programmer. Data in the catalogue were sufficient to pick PXI-4461 as the DAC device to drive the coils producing the ELF magnetic field pulses. However, additional measurements were required to determine, whether ADC in this device is sufficient to acquire the signal of the ^{199}Hg comagnetometer. PXI-5922 ADC is claimed to have better parameters (sampling frequency and noise levels), and was thus picked as an alternative in case of ADC in PXI-4461 being unsatisfactory.

Measurements were performed exactly in the already described manner. PXI-4461 generated signal on two channels, and it was recorded by either PXI-4461 or PXI-5922. Both devices had their clock synchronised to an atomic clock via the `PXI_CLK10` bus. Results are presented in Table 4.2.

device	**upper limit on a frequency drift (nHz)**	**upper limit on RMS noise (µV)**	
PXI-4461	0.2	34	(Fig. 4.11)
PXI-5922	3.2	26	(Fig. 4.12)

TABLE 4.2: *Results of measurements performed with the use of two ADC modules.*

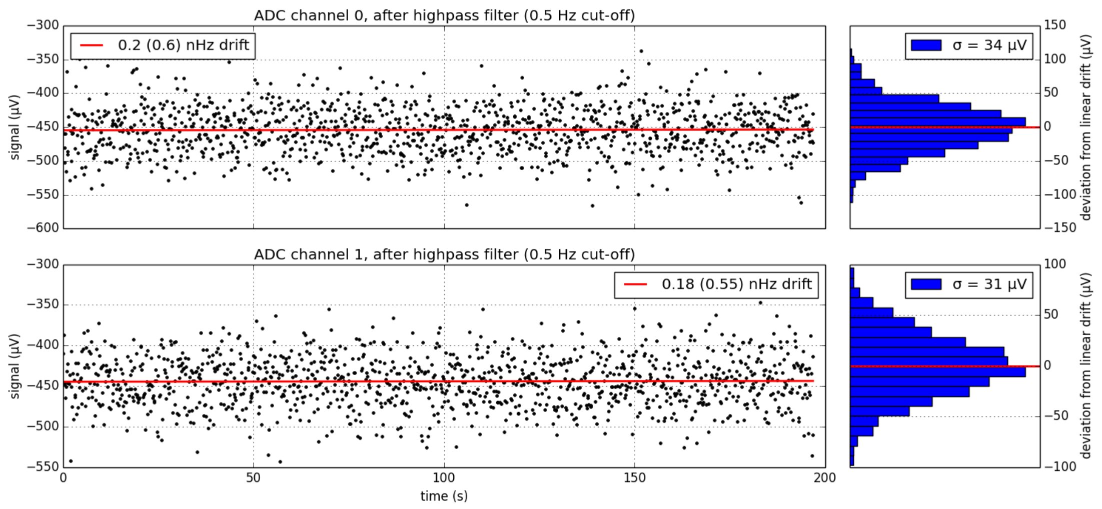


FIGURE 4.11: *Results of frequency stability measurement of the National Instruments* ***PXI-4461****. See Fig. 4.7 for explanation. No frequency drift was measured within uncertainty of 0.6 nHz.*

The PXI-5922 device, as claimed by the manufacturer, performs slightly less noisy measurements. Both devices, when synchronised to an atomic clock, provide an excellent frequency stability of generated signal – on a nHz level. Based on the results, it was decided that one PXI-4461 module can serve as both ADC and DAC in the experimental set–up.

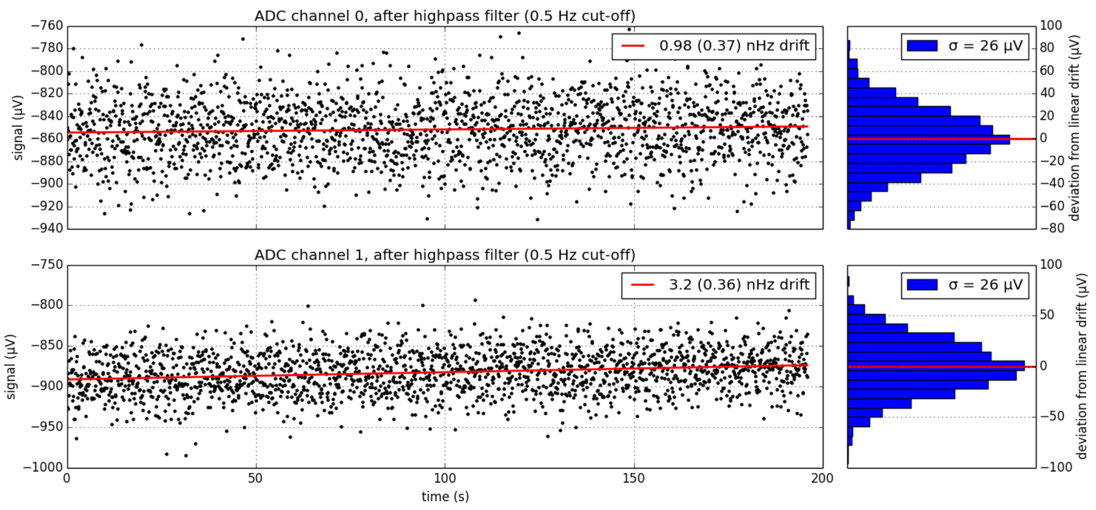


FIGURE 4.12: *As Fig. 4.11, but for the **PXI-5922** module. The measured frequency is 3 nHz.*

4.6.5 Timing accuracy of TTL pulses

In the current version of the software, the TTL pulses have only a milisecond accuracy, as they are driven by the *general scheduler*. It is, however, possible to make them more accurate by writing a dedicated scheduling mechanism (which is, in fact, planned). The purpose of presented measurement was to check how accurate TTL pulses would be, if such mechanism were used.

The most accurate timing in the system is provided by the DAC output, so it was used as a reference. A very simple, hard–coded *dedicated scheduling mechanism* was embedded into the software. The system was programmed, so that two events should happen simultaneously: change of the TTL line state and a jump in a voltage on the DAC output. Both were measured on an oscilloscope, which gave result presented in Fig. 4.13. The system showed to have ≈ 50 μs timing accuracy and ≈ 30 μs time jitter. These are expected results, as they correspond to timing properties of the real–time operating system hosting the software. However, it must be stressed that until a dedicated scheduling mechanism is implemented, accuracy remains in millisecond regime.

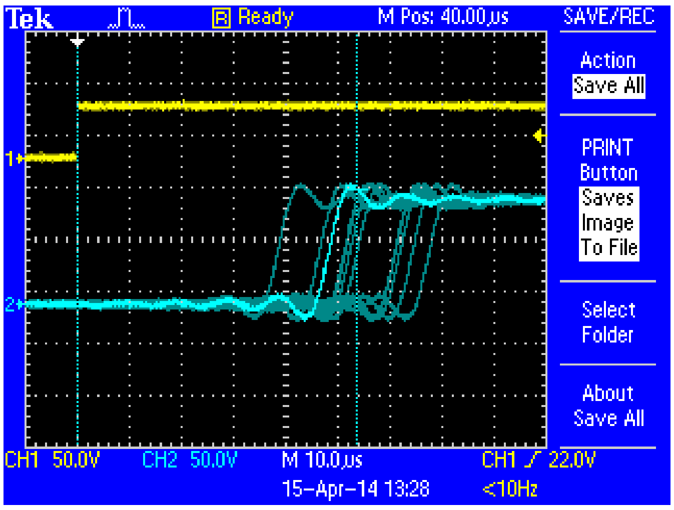


FIGURE 4.13: *Result of TTL timing accuracy in the PXI system measurement. Yellow line is TTL pulse (used as trigger in the scope), blue – ADC pulse. Several runs are presented one on top of another. The system was configured such, that jumps on both lines should happen simultaneously. The delay between the jump is timing accuracy* ($\approx 50\,\mu s$)*. The spread of the blue line corresponds to time jitter of the system* ($\approx 30\,\mu s$)*.*

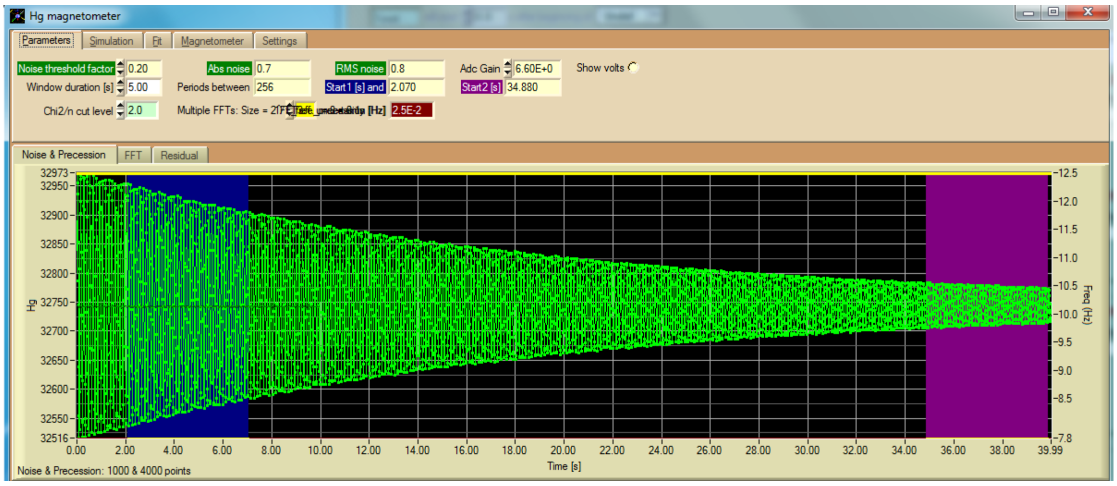
4.6.6 On–site test

Being aware of possible unforeseen troubles, that would not arise until the PXI system is actually plugged into the real apparatus, an on–site test was performed. It took place on February 2014 at the Paul Scherrer Institute in Villigen, Switzerland — where the nEDM experiment is located. The PXI system was set up instead of the normal control and DAQ system. The goal was to perform the *mercury cycles*, that is to repeatedly:

1. Fill the storage chamber with polarised ^{199}Hg atoms.
2. Rotate their spins with an ELF magnetic field pulse.
3. Continuously read voltage on the photomultiplier, in order to record precession of the mercury atoms.
4. Pump the system down to prepare for the next cycle.

The test was qualitative. Only behaviour was tested, not exact performance, because the PXI system was equipped only with basic modules that were used for development.

The test indeed uncovered several faults, but all were fixed during five days that were designated for it. By the end, the system was able to successfully run *mercury cycles* overnight. Results of one cycle, as seen on the interface of the AcqEDM program, is presented in Fig. 4.14.


Noise threshold factor 0.20
Abs noise 0.7
RMS noise 0.8
Adc Gain 6.60E+0
Show volts
Window duration [s] 5.00
Periods between 256
Start1 [s] and 2.070
Start2 [s] 34.880
Chi2/n cut level 2.0
Multiple FFTs: Size = 2 [illegible] [Hz] 2.5E-2

Noise & Precession: 1000 & 4000 points

FIGURE 4.14: *Recorded* ^{199}*Hg comagnetometer signal in a cycle performed with the PXI system used for both control and data acquisition. An FID signal is visible, proving that the cycle did run successfully.*

4.6.7 Summary

It was shown that devices in the PXI crate can have their clocks synchronised to a reference frequency source via the common `PXI_CLK10` bus. An atomic clock needs to be plugged in the back of the PXI chassis and each device needs to be configured to use the bus as a reference. As a very good solution covering both ADC and DAC, National Instruments PXI-4461 module was selected and its parameters were confirmed to be satisfactory. It was also shown, that the system is capable of controlling TTL lines with accuracy $\approx 50\,\text{ms}$ (after additional software improvements). Most importantly, the system was able to successfully work with the experimental apparatus.

Chapter 5

Summary

Analysis of the ^{199}Hg comagnetometer FID signal

Several methods of analysis of the comagnetometer FID signal were proposed and discussed. After the presented research, the traditional *two windows* approach is considered by the author best suited for the analysis. However, it was pointed out that the sizes of the windows should be optimised and, in particular, should not be equal. The optimisations performed for the real data (run 7650, $\tau \approx 40\,\mathrm{s}$) resulted in sizes 6.5 and 130 s, which are much different from the ones usually used: 15 and 15 s. The optimised *two windows* method offered in this case a factor 10 improvement in precision, and a factor 4 in accuracy.

Furthermore, another approach to the analysis of the comagnetometer FID signal was proposed, following the idea presented in the original work by Chibane et al. [6]. In the proposed approach, the FID signal is first translated into a cumulative phase array, and only the result is used to estimate the average frequency. Such division facilitates the analysis by encapsulating the two problems. At the same time, it does not restrict achievable precision, nor accuracy. In fact, the *two windows* approach gives almost always the same results, regardless whether it is implemented in the traditional way (two waves fitted directly to the signal), or in the proposed new way (two straight lines fitted to the cumulative phase array). There are, however, cases when the traditional implementation either gives a wrong result or does not converge at all. This happens for signals with a poor signal–to–noise ratio or a short relaxation time. The implementation in the new approach, on the other hand, gives correct results even then. For this reason, in addition to the mentioned clarification of the analysis, I consider the new approach superior.

Last, but not least, a practical way of estimation of systematic errors in the analysis was presented. This includes systematics caused by frequency drifts, as well as by the filter. In order to do this, a run is first tentatively analysed to assess parameters of the signal:

signal–to–noise ratio, relaxation times and magnitude of frequency drifts. These values are then used to generate signals with similar parameters, but exactly known frequencies. These signals allow one to perform the mentioned optimisation of the windows sizes in the *two windows* approach, as well as to estimate systematic and statistical uncertainty of the frequency estimation.

The PXI–based DAQ and control system

A prototype of a new DAQ and control system for the nEDM experiment has been developed. The created solution offers a rapid development cycle and extensive scripting capabilities. It also follows the DAQ scheme proposed for the next generation of the experiment (n2EDM).

Quantitative tests of ADC, DAC and timing capabilities of the system were performed. The results of the tests show, that the system fulfils the requirements for serving as a control system in the experiment. The prototype also proved that it is already capable of working together with the experimental apparatus.

Appendix A

The `hg_toolkit`

A.1 The choice of the programming language

There are many very popular scientific suites with dedicated syntax, most notably *Matlab* and *Mathematica*. These programming languages are from the core optimised for writing scientific programs. Therefore, their syntax strongly supports a use of specific for science–programming concepts, like arrays and matrices. This makes most of the statements very compact and natural. However, such programming languages have at the same time huge limitations when it comes to large projects and writing reusable code.

A different approach is taken by some frameworks. The alternative concept is to choose an existing, well established programming language and create a set of scientific tools in form of a library. The most notable representatives are ROOT with C++ as the base language and SciPy (Scientific Python) with Python. Programs created in such frameworks can benefit from a whole range of possibilities that their base language offers, which are incomparable with what is offered by languages which are only science–oriented. As both C++ and Python are used worldwide in industry, they highly support large projects with reusable code. The downside of this approach, that the scientific library is still constrained by the syntax of the base language.

For writing the `hgtoolkit` SciPy was chosen. The project from the very beginning was meant to be a well–structured, reusable set of tools for the analysis of the comagnetometer signal. This immediately pointed towards either SciPy or ROOT. The former was chosen, because SciPy uses familiar concepts from Matlab, often even keeping the same functions names. At the same time it is based on Python, which is often considered the most friendly existing programming language (as opposed to ROOT's C++). ROOT is much more performance–oriented, which is not crucial in the analysis of the comagnetometer signal.

```
"""Demonstrate the two windows method.
"""

from pylab import *

import hg_toolkit.simulation as simulation
from hg_toolkit.methods.two_windows import two_windows

with_filter = False

T, D, sD, freal, sfCR = simulation.rand_poly_frequency_two_exp_amplitude(
    i=9,
    seed_drift=3242,
    deg=3,
    CRbound=True,
    with_filter=with_filter)

f, sf, Tf, F, sF = two_windows(
    T=T,
    D=D,
    sD=sD,
    submethod='phase',
    prenormalize=False,
    double_exp=(True, False),
    phase_at_end=True,
    win_len=(15 / 20., 15),
    verbose=True)

print('sf:      ', sf)
print('CR bound:', sfCR)
print('delta:   ', (f - freal) / sf)

show()
```

Listing A.1: A full listing of the script demonstrating the two windows method: `hg_toolkit/scripts/methods/two_windows/demo.py`

A.2 Structure of the toolkit

The toolkit itself has a form of a python package, divided further into modules. This means that once it is recognised by the system, it can be loaded into any Python script or an interactive session.

Together with the package, a large set of scripts is provided. They all use the toolkit to make a range of things: from a simple demonstration of a method of frequency estimation, to a full analysis of a real data set. A listing of an example script, demonstrating the *two windows* method, is shown in Listing A.1.

Also, a full list of files bundled with the toolkit, both the package and the scripts, is presented in Fig. A.1.

```
hg_toolkit
├── noise_from_data.py
├── README.txt
├── scaling.py
├── hg_toolkit ........ the actual toolkit in form of a Python package
│   ├── cramer_rao.py ........ calculation of the Cramer–Rao bound
│   ├── cycle_iterator.py ........ an utility iterator cycling through cycles in a data–file
│   ├── flter.py ........ a module implementing the filter (Sec 3.8)
│   ├── frequency_in_windows.py
│   ├── Hg_offline_lib.py ........ a set of utility functions
│   ├── implementation_test.py
│   ├── methods_tests.py ........ a set of functions to automatically test methods of frequency estimation
│   ├── scan_parameter.py ........ a tool to scan a method's parameter, eg. size of windows (Sec. 3.9.4)
│   ├── simulation.py ........ a Python module implementing simulation of the comagnetometer signal
│   ├── total_phase.py ... a function to find a total phase difference by counting zero–crossings (Sec. 3.9.2)
│   └── methods ........ methods of average frequency estimation
│       ├── direct_fit.py ........ described in Sec. 3.6
│       ├── fit_to_every_period.py
│       ├── many_windows.py
│       ├── point_by_point.py ........ described in Sec. 3.10
│       ├── poly_fit.py
│       ├── two_windows.py ........ described in Sec. 3.9
│       └── windowed_phase.py ........ described in Sec. 3.11
└── scripts ........ a set of example scripts that use the toolkit
    ├── filter_demo.py ........ demonstrates working of the filter
    ├── amplitude_oscillations ........ Scripts investigating the oscillation of amplitude (Sec. 3.12)
    │   ├── analyse_one_cycle_and_dump_spline.py
    │   ├── plot_across_cycles.py
    │   └── shift_test.py
    ├── data_analysis ........ scripts to perform analysis of the real data (3.13)
    │   ├── find_drift_in_run.py
    │   ├── plot_example_signal.py
    │   ├── run_analysis.py
    │   └── tau_and_StN_distribution_in_run.py
    └── methods ........ scripts dedicated to a particular method of frequency estimation
        ├── methods_comparison.py ........ compare methods on a precission–accuracy plane (eg. Fig. 3.23)
        ├── direct_fit
        │   ├── demo.py ........ demonstrate the method on an example signal
        │   └── test_implementation.py ........ run test described in Sec. 3.6.2
        ├── fit_to_every_period ........ described in Sec. 3.6
        │   ├── demo.py
        │   └── test_implementation.py
        ├── many_windows
        │   ├── demo.py
        │   └── test_implementation.py
        ├── point_by_point ........ described in Sec. 3.10
        │   ├── accuracy_with_filter.py
        │   ├── amplitude_oscillations.py ..... test influence of the amplitude oscillations on the method
        │   ├── demo_poly_fit.py
        │   ├── demo_two_windows.py
        │   ├── test_implementation_poly_fit.py
        │   ├── test_implementation_two_windows.py
        │   └── two_windows_optimization.py ........ optimise size of the windows, see (Sec. 3.9.4)
        ├── two_windows ........ described in Sec. 3.9
        │   ├── accuracy_with_filter.py
        │   ├── demo.py
        │   ├── first_window_smaller.py
        │   ├── test_implementation.py
        │   └── windows_sizes_optimization.py ........ optimise size of the windows, see (Sec. 3.9.4)
        └── windowed_phase ........ described in Sec. 3.11
            ├── adaptive_chi2_filtered_and_not.py
            ├── adaptive_chi2_limit_optimization.py
            ├── adaptive_chi2_windows_size_optimization.py
            ├── demo_adaptive_chi2.py
            ├── demo_poly_fit.py
            ├── demo_two_windows.py
            ├── poly_fit_degree_optimization.py
            ├── test_implementation_adaptive_chi2.py
            ├── test_implementation_poly_fit.py
            └── test_implementation_two_windows.py
```

FIGURE A.1: *A list of files that are bundled with the* `hgtoolkit`.

Appendix B

Full list of implemented PXI commands

command	parameter(s)	description
`UTIMer`		
· `:ID?`		reply with subsystem's name: `PSI_nEDM_PXI_MR`
· `:TTL`		
· · `:TTL##`	1|0|PULSE	# is a number form 0 to 7; set to high, low, or send a pulse
· · `:TTL##?`		returns current state of the TTL channel #
· `:ELF`		
· · `:SAMPling`		integer sampling for ELF signals in samples per second
· · `:SAMPling?`		
· · `:ELF##`		# is a number form 0 to 3
· · · `[:SEND]`	[float]	optional argument is duration - if none given corresponding `:DURation` is used
· · · `:FREQuency`	float(Hz)	
· · · `:FREQuency?`		
· · · `:AMPLitude`	float(V, peak-to-peak)	
· · · `:AMPLitude?`		
· · · `:ENVelope`	SQUAre|SINE	
· · · `:ENVelope?`		
· · · `:DURation`	float(s)	
· · · `:DURation?`		
· · · `:PHASe`	float(deg.)	
· · · `:PHASe?`		
· · · `:HardwareOUTput`	integer	On which hardware channel is this ELF pulse output. If there are several ELFs being generated simultaneously on same channel, they are added together.
· · · `:HardwareOUTput?`		

Continued on next page...

command	parameter(s)	description
· `:ADC`		
· · `:SAMPling`	float(S/s)	
· · `:SAMPling?`		
· · `:STARt`	0-7	number of buffer where data will be loaded; there are eight buffers numbered 0-7
· · `:STOP`		
· · `[:MEASure]`	0-7 float(s)	`"UTIM:ADC:MEAS 0 2"` means start measuring now, put data in buffer 0 and stop measuring in two seconds
· · `:SEND`	0-7	send buffer on data port. The reply on data port will first contain ASCII header `"UTIM:ADC:SEND"`, then a binary unsigned long - number of data points, then a binary array of unsigned shorts
· · `:DC`		take DC measurement and send result on data port. The reply will be ASCII encoded `"UTIM:ADC:DC 3.422342"`, where the number is a decimal-coded float - DC value in volts.
· `:TRIGger`		
· · `:REQuest`		request trigger and set TRIG:MASK to 1
· · `:FIRE`		manually fire trigger, sets TRIG:MASK to 0
· · `:MASK`	1 or 0	1 - start schedule on trigger, 0 - ignore trigger
· · `:CAME?`		query if the trigger came (resets after :TRIGger:REQuest)
· · `:TIME?`		return time of last trigger (or manual fire) in format: `2014-01-25_14:12:37.687_UTC`
· `:SCHedule`		
· · `[:SET]`	for example: `"0.0 UTIM:SCH:STEP 1; 1.0 UTIM:VAT 1; 2.0 UTIM:ELF:NEUT; ..."`	set schedule; schedule description should be enclosed in double quotes; schedule is a semicolon–separated list of space–separated pairs (time, command).
· · `[:SET?]`		reply with currently set schedule
· · `:STEP`	string|integer	set current step (either name or number)
· · `:STEP?`		reply with current step
· · `:END`		mark end of schedule
· · `:END?`		query end of schedule (resets at schedule start)
· · `:STTL`	string|integer	equivalent to `"UTIM:TTL:TTL1 PULSE; UTIM:STEP string|integer"`

Bibliography

[1] Internal nedm collaboration materials. http://nedmpsi.atlassian.net/.

[2] H. G. Dehmelt. Modulation of a light beam by precessing absorbing atoms. *Physical Review*, X(1):1924–1925, 1957.

[3] M. Fertl. *A laser based mercury co-magnetometer for the neutron electric dipole moment search*. PhD thesis, Technische Universität München, 2013.

[4] M. Perkowski. Optimization of the ^{199}hg comagnetometry for the nedm experiment. Master's thesis, Jagiellonian University, Kraków, 2012.

[5] M. Burghoff. Progress report. PSI Proposal R-05-03.1, 2013.

[6] Y Chibane, S K Lamoreaux, J M Pendlebury, and K F Smith. Minimum variance of frequency estimations for a sinusoidal signal with low noise. *Measurement Science and Technology*, 6(12):1671, 1995. URL http://stacks.iop.org/0957-0233/6/i=12/a=004.

[7] K. Green, P.G. Harris, P. Iaydjiev, D.J.R. May, J.M. Pendlebury, K.F. Smith, M. van der Grinten, P. Geltenbort, and S. Ivanov. Performance of an atomic mercury magnetometer the neutron edm experiment. *Nuclear Instruments and Methods in Physics Research A*, 404:381–393, September 1997. URL http://dx.doi.org/10.1016/S0168-9002(97)01121-2.

[8] J. Zenner. *The search for the neutron electric dipole moment*. PhD thesis, Johannes Gutenberg–Universität, Mainz, 2013.

[9] Jacek Zejma. Private communication, 2014.

[10] D. Rife and R. Boorstyn. Single tone parameter estimation from discrete-time observations. *IEEE Transactions on Information Theory*, 20, 1974. doi: 10.1109/tit.1974.1055282. URL http://libgen.org/scimag/index.php?s=10.1109/tit.1974.1055282.

[11] R. P. Brent. Algorithms for minimization without derivatives. 1973.

[12] Georg Bison and W. Heil. Private communication, 2014.

[13] O. Bourrion, G. Pignol, D. Rebreyend, and C. Vescovi. Development of a multifunction module for the neutron electric dipole moment experiment at {PSI}.

Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 701(0):278 – 284, 2013. ISSN 0168-9002. doi: http://dx.doi.org/10.1016/j.nima.2012.10.077. URL http://www.sciencedirect.com/science/article/pii/S0168900212012284.

[14] Jacek Zejma. *Experimental Quest for the Neutron Electric Dipole Moment.* habilitation thesis, Jagiellonian University, 2014. ISBN 978-83-60391-76-1.

[15] Parts of a pxi system. http://www.ni.com/white-paper/4811/en/. Accessed: 2014.05.07.

[16] Georg Bison. Private communication, 2014.